Oracle® Scripting

Concepts and Procedures

Release 11*i*

August 2000

Part No. A86105-01

Contents

Send Us Your Comments

Oracle Scripting Concepts and Procedures, Release 11*i*

Part No. A86105-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us via the postal service.

Oracle Corporation
CRM Content Development Manager
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Welcome to the Oracle Customer Relationship Management, Release 11*i*, suite of applications.

This Concepts and Procedures provides information and instructions to help you work effectively with Oracle Scripting.

This preface explains how Concepts and Procedures is organized and introduces other sources of information that can help you.

Intended Audience

This guide is aimed at the following users:

- Technical Service Representatives (TSR)
- Customer Service Representatives (CSR)
- System Administrators (SA), Database Administrators (DBA), and others with similar responsibility
- Call Center Managers and Supervisors

This guide assumes you have the following prerequisites:

1. Understanding of computer-telephony integration (CTI)
2. Understanding of call center technology
3. Understanding of the company business processes
4. Understanding of Oracle databases
5. Background in SQL and Java programming

Structure

This manual contains the following sections:

“Understanding Oracle Scripting" provides overviews of the application and its components, explanations of key concepts, features, and functions, as well as the application’s relationships to other Oracle or third-party applications.

“Using Oracle Scripting” provides process-oriented, task-based procedures for using the application to perform essential business tasks.

“Administering Oracle Scripting" provides task-based procedures for required for ongoing system maintenance and includes information on administration tools and utilities.

“Customizing Oracle Scripting” provides information on using Best Practice Scripts to develop scripts.

Related Documents

For more information, see the following manuals:

- *Oracle Call Center Applications Setup*
- *Oracle Scripting Implementation Guide*
- *Oracle Scripting Technical Reference Manual*

Understanding Oracle Scripting

This topic group provides overviews of the application and its components, explanations of key concepts, features, and functions, as well as the application's relationships to other Oracle or third-party applications.

This topic group includes the following topics:

What is Oracle Scripting?

Oracle Scripting is a three-tier 100% pure Java application that presents a navigable script to the desktop of an interaction center agent. A script is a dialog or guide for agent-customer interactions.

An Oracle Scripting script is a series of panels which appear one at a time at the agent desktop. Each panel prompts the agent for a response. The agent navigates through the script by selecting an option in the panel.

An Oracle Scripting script can also manage complex processing. Objects, which are invisible to the agent, can be inserted into the script workflow. As the agent moves effortlessly through uncomplicated script panels, Oracle Scripting objects embedded in the script control complicated processing, such as rules-based branching and data integration.

The Building Blocks of a Script Workflow

A script workflow is made up of objects connected by branches. Objects are processing units that tell the script what to do. Branches are processing units that tell the script where to go.

There are three objects in Oracle Scripting Author:

- Panels
- Groups
- Blocks

See also: Getting Started with Oracle Scripting Author

Panels

A panel object contains the information that is displayed in Oracle Scripting Engine at runtime. Answer controls in the panel object are used to accept information from the user and advance the script workflow.

References

Defining the Panel Name

Defining the Panel Label

Defining Panel Pre-Actions

Defining Panel Post-Actions

Defining Panel Text

Substituting a Java Bean for a Panel

Groups

A group object is a container for objects. Groups are used to:

- logically group a section of the script workflow
- create a shortcut button that jumps to a section of the script workflow at runtime
- restrict the access of certain or all users to a section of the script workflow at runtime

In addition, when a saved script workflow is imported into a current script project, it is imported as a group.

References

Inserting a Group

Importing a Saved Script as a Group

Defining the Group Name

Defining Group Pre-Actions

Defining Group Post-Actions

Defining Group Permission

Blocks

A block is used to query, update, or insert information in a database. A block is always associated with one or more database tables.

A block can also be a container for panel objects. The answer controls in the panel objects can be used in block processing.

References

Inserting a Block

Defining the Block Name

Defining Block Pre-Actions

Defining Block Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Defining a Database Update Block

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Branches

A branch is used to connect objects. The type of branch determines the next object in the script workflow. There are four types of branches:

- **Default:** The destination is the designated object. If there is more than one branch, the default branch is used if no other branch is valid.
- **Distinct:** The destination is based on the answer selected in a panel object.
- **Conditional:** The destination is based on the evaluation of a Boolean expression.
- **Indeterminate:** The destination is based on information obtained at runtime.

References

Defining a Default Branch

Defining a Distinct Branch

Defining a Conditional Branch

Defining an Indeterminate Branch

Defining the Branch Name

Reordering Branches

Using Oracle Scripting Author

This topic group provides process-oriented, task-based procedures for using the application to perform essential business tasks.

This topic group includes the following topics:

Getting Started with Oracle Scripting Author

1. Start a new script project.
2. If you want to logically group a section of the script workflow, access a section of the script workflow by a shortcut button or restrict access to a section of the script workflow at runtime, then do the following:

   a. Insert a group.

   b. Define the following group properties:

Group Property	Required?	Description
Name	Yes	The group name is how the group is identified in Oracle Scripting Author.
Pre-Action	No	The pre-action takes place before the group is executed at runtime.
Post-Action	No	The post-action takes place after the group is executed at runtime.

   c. Drill down into the group.

   d. If, in the group, you want to query, update, or insert information in database tables, then continue to step 4.

   e. If, in the group, you want to display information (such as text, an image, a hyperlink, a variable, or a Java bean) to an agent or accept information entered by an agent, then continue to step 5.

   Note: If the Start object is the only object in the group, you must Insert a termination point and draw a default branch between the Start object and the termination point.

3. If you want to use a saved script workflow in the current script project, then import a saved script project.

   The script workflow is imported as a group in the current script project.
4. If you want to query, update, or insert information in database tables, then do the following:

   a. Insert a block.

Note: An update block must be preceded by a query block in the script workflow.

---

b. Define the following block properties:

Block Property	Required?	Description
Name	Yes	The block name is how the block is identified in Oracle Scripting Author.
Pre-Action	No	The pre-action takes place before the block is executed at runtime.
Post-Action	No	The post-action takes place after the block is executed at runtime.

c. Define the parameters for the data source.

d. Drill down into the block.

e. If, in the block, you want to display information (such as text, an image, a hyperlink, a variable, or a Java bean) to an agent or accept information entered by an agent, then continue to step 5.

Note: If the Start object is the only object in the block, you must Insert a termination point and draw a default branch between the Start object and the termination point.

---

5. If you want to display information (such as text, an image, a hyperlink, a value, or a Java bean) to an agent or accept information entered by an agent, then do the following:

   a. Insert a panel.

   b. Define the following panel properties:

Panel Property	Required?	Description
Name	Yes	The panel name is how the panel is identified in Oracle Scripting Author.
Label	Yes	The panel label is how the panel is identified at runtime.

Panel Property	Required?	Description
Pre-Action	No	The pre-action takes place before the panel is executed at runtime.
Post-Action	No	The post-action takes place after the panel is executed at runtime.

c. If you want to substitute a Java bean for a panel, then define the Java bean.
d. If you want to display information in the panel, then define the panel text.
e. Define the controls that the agent use to enter information or advance the script workflow at runtime.
f. If you want to add another panel, then repeat steps a through e.
g. Insert a termination point.

6. To construct the script workflow, do the following:
   a. Draw branches between the objects to indicate the flow of the script.
   b. Define the following branch properties for each branch.

Branch Property	Required?	Description
Name	Yes	The branch name is how the branch is identified in Oracle Scripting Author.

   c. Define the parameters for each branch.
   d. Optionally, align the objects and branches.
7. Define the script pre- and post-actions.
8. Layout the script header.
9. Define the script shortcut buttons.
10. Program the script disconnect button.
11. Define the script runtime attributes.
12. Check the syntax of the script.
13. Save the script.
14. Deploy the script.

Working with Script Files

You can perform the following tasks:

Starting a New Script Project

Opening an Existing Script Project

Saving a Script Project

Reversing All Changes Since the Last Save Command

Reversing All Changes Since the Last Open Command

Importing a Script Workflow

Exporting a Script Workflow

Printing a Script Workflow

Closing a Script Project

Starting a New Script Project

Use this procedure to create a new script.

Prerequisites

None

Steps

- Choose **File** > **New**.

  The Start object appears in the workspace.

Opening an Existing Script Project

Use this procedure to open a saved script.

Prerequisites

None

Steps

1. Choose **File** > **Open**.

2. In the Open window, locate the script (*.script file) that you want to open and then click **Open**.

   The script objects appear in the workspace.

Saving a Script Project

Use this procedure to save a script. You can save a script using its current name and location, or save a copy of the script using a different name and location.

> **Note:** The asterisk to the right of the window title for Oracle Scripting Author indicates that changes have been made to the script that have not been saved.

Prerequisites

None

Steps

1. Do one of the following:
   - To save the script using its current name and location, choose **File** > **Save**.
   - To save a copy of the script using a different name and location, choose **File** > **Save As**.

Reversing All Changes Since the Last Save Command

Use this procedure to undo all changes made since you last saved the script. This action can be performed any time script changes are made and saved.

Prerequisites

None

Steps

- Choose **File** > **Revert to > Last Save**.

Importing a Script Workflow into a Script Project

Use this procedure to import a saved script workflow into the current script project.

Prerequisites

None

Steps

1. Choose **File > Import**.

   The Open dialog box appears.

2. Locate the script file (*.script) that you want to import.
3. Click the file name and then click **Open**.

   The imported script workflow appears in the workspace as a group.

Exporting a Script Workflow as a Separate Script File

Use this procedure to export a group as a separate script file.

Prerequisites

None

Steps

1. In the workspace, select a group.
1. Choose **File > Export**.

   The Save dialog box appears.

2. Select the destination.
3. Type the file name and then click **OK**.

Printing a Script Workflow

Use this procedure to print a script.

Prerequisites

None

Steps

1. Choose **File > Print**.
2. Select your print options and then click **OK**.

Closing a Script Project

Use this procedure to close a script. You can close a script project at any time. If you have unsaved changes, then you'll be asked whether you want to save the changes.

Prerequisites

None

Steps

- From the **File** menu, choose **Close**.

Working with the Tool Palette

You can perform the following tasks:

Inserting an Object

Inserting a Branch

Selecting Objects

Deselecting Objects

Moving Objects

Moving Branches

Deleting Panels, Groups, and Blocks

Deleting Branches

Inserting an Object

You can perform the following tasks:

Setting the Properties Dialog Box to Pop Up at Object Creation

Inserting a Panel

Inserting a Group

Inserting a Block

Inserting a Termination Point

Setting the Properties Dialog Box to Pop Up at Object Creation

Use this procedure to set up Oracle Scripting Author to automatically display the Properties dialog box when you insert a panel, group, or block into the workspace.

Prerequisites

None

Steps

- Choose **View** > **Popup on Blob Creation**.

Inserting a Panel

Use this procedure to insert a panel into the workspace.

Prerequisites

None

Steps

1. In the tool palette, click the **Panel Insertion Mode** tool.

   When the pointer is over the workspace, the pointer is a pointing finger

2. In the workspace, click where you want to insert the panel.

   If the Popup on Blob Creation option is selected, then the Properties dialog box for the object appears.

References

Setting the Properties Dialog Box to Pop Up at Object Creation

Defining the Panel Name

Defining the Panel Label

Defining Panel Pre-Actions

Defining Panel Post-Actions

Defining Panel Text

Substituting a Java Bean for a Panel

Inserting a Group

Use this procedure to insert a group into the workspace.

Prerequisites

None

Steps

1. In the tool palette, click the **Group Insertion Mode** tool.

   When the pointer is over the workspace, the pointer is a pointing finger

2. In the workspace, click where you want to insert the group.

   The **Go down into child graph** button in the workspace toolbar is made active. If the Popup on Blob Creation option is selected, then the Properties dialog box for the object appears.

References

Drilling Up or Down to a Related Script

Setting the Properties Dialog Box to Pop Up at Object Creation

Importing a Saved Script as a Group

Defining the Group Name

Defining Group Pre-Actions

Defining Group Post-Actions

Defining Group Permission

Inserting a Block

Use this procedure to insert a block into the workspace.

Prerequisites

None

Steps

1. In the tool palette, click the **Block Insertion Mode** tool.

   When the pointer is over the workspace, the pointer is a pointing finger

2. In the workspace, click where you want to insert the block.

   The **Go down into child graph** button in the workspace toolbar is made active. If the Popup on Blob Creation option is selected, then the Properties dialog box for the object appears.

References

Drilling Up or Down to a Related Script

Setting the Properties Dialog Box to Pop Up at Object Creation

Inserting a Termination Point

Use this procedure to insert a termination point into the workspace.

Note: Every script and all subscripts for groups and blocks must end with a termination point. Even if the block does not have any panels, it still must have a termination point.

Prerequisites

None

Steps

1. In the tool palette, click the **Termination Point Insertion Mode** tool.

   When the pointer is over the workspace, the pointer a pointing finger

2. In the workspace, click where you want to insert the termination point.

Inserting a Branch

You can perform the following tasks:

Setting the Properties Dialog Box to Pop Up at Branch Creation

Inserting a Straight Branch

Inserting a Branch with Corners

Adding a Corner to an Existing Branch

Moving a Corner of a Branch

Deleting a Corner from a Branch

Setting the Properties Dialog Box to Pop Up at Branch Creation

Use this procedure to set up Oracle Scripting Author to automatically display the Properties dialog box when you insert the branch into the workspace.

Prerequisites

None

Steps

- Choose **View** > **Popup on Branch Creation**.

Inserting a Straight Branch

Use this procedure to insert a straight line branch with no corners.

Prerequisites

To insert a branch, you must have at least two object in the workspace.

Steps

1. In the tool palette, click a branch tool.

   When the pointer is over the workspace, the pointer is a cross hair (+).

2. In the workspace, drag the pointer from the source object to the destination object.

   When you release the pointer over the destination object, the branch arrow appears. The branch is red. This indicates that the branch is currently selected.

References

Setting the Properties Dialog Box to Pop Up at Branch Creation

Inserting a Branch with Corners

Use this procedure to insert a branch with corners.

Prerequisites

To insert a branch, you must have at least two object in the workspace.

Steps

1. In the tool palette, click a branch tool.

   When the pointer is over the workspace, the pointer is a cross hair (+).

2. In the workspace, drag the pointer from the source object to the desired location of the first corner and release the pointer.

3. Move the pointer to the desired location of the next corner and click. The line is drawn automatically.

4. Click the destination object.

   When you click destination object, the branch arrow appears. The branch is red. This indicates that the branch is currently selected.

References

Setting the Properties Dialog Box to Pop Up at Branch Creation

Adding a Corner to an Existing Branch

Use this procedure to add a corner to an existing branch.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.

2. In the workspace, select a branch.

   When the branch is selected, the branch is red.

3. Drag the branch to the desired location and then release the pointer.

   When you release the pointer, a corner is created on the branch.

Moving a Corner of a Branch

Use this procedure to move a corner of a branch.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a branch.
3. Select a branch corner.

   When the pointer is over the branch corner, the pointer is a move icon. When the branch corner is selected, the branch is no longer red.
4. Drag the branch to the desired location and then release the pointer.

Deleting a Corner from a Branch

Use this procedure to delete a corner from a branch.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a branch.
3. Select a branch corner.

   When the pointer is over the branch corner, the pointer is a move icon. When the branch corner is selected, the branch is no longer red.
4. Choose **Edit** > **Delete**.

Selecting Objects

Use this procedure to select one or more objects. You can select several objects at the same time, or you can add objects to an existing selection.

Prerequisites

All of the objects to be selected must be in the same workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. Do one of the following:
   - To select an object, click the object.
   - To select several object in the same area, point outside the objects and drag diagonally to draw a selection border around them.

     All objects in or on the selection border are selected. This includes branches.
   - To add an object to a selection, Shift-click the object. You can also Control-click the object.

Deselecting Objects

Use this procedure to deselect one or more selected objects.

Prerequisites

None

Steps

Do one of the following:

- To deselect an object, click outside the object.
- To deselect one of several selected objects, Shift-click the object. You can also Control-click the object.
- To deselect all selected objects, click in the workspace away from any objects.

Moving Objects

Use this procedure to move panels, groups, and blocks in the workspace.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.

2. In the workspace, select one or more objects.
3. Drag the object to the desired location.

   If a branch is connected to the object, then the branch moves with the object.

Moving Branches

Use this procedure to change the destination object for a branch. You cannot move the start or termination point of a branch independently of a object.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select the destination object for a branch.

   Note: This procedure requires you to delete the destination object. To retain the object, use the Edit command to copy the object and paste it in the workspace.

3. Choose **Edit** > **Delete**.
4. Select the new destination object.
5. Drag the destination object to of the branch.
6. When the branch is red, release the destination object.

   The branch is redrawn to the new destination object.

Deleting Panels, Groups, and Blocks

Use this procedure to delete objects from the workspace.

Note: Deleting an object deletes all properties, edges (outgoing branches), and subgraphs associated with the object. To save a group as an independent script file, see Exporting a Script.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select one or more objects.
3. Choose **Edit** > **Delete**.

Deleting Branches

Use this procedure to delete branches from the workspace.

Note: Deleting a branch deletes all properties associated with the branch.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a branch.

   When the branch is selected, the branch is red.
3. Choose **Edit** > **Delete**.

Viewing Objects in the Workspace

You can perform the following tasks:

Fitting a Script in the Workspace

Drilling Up or Down to a Related Script

Aligning Objects

Fitting a Script Layout in the Workspace

You can zoom in to focus on the details of the script layout or zoom out to see more of the script layout.

Prerequisites

None

Steps

Do one of the following:

- To increase the magnification, click the **Zoom In** button in the workspace toolbar.
- To decrease the magnification, click the **Zoom Out** button in the workspace toolbar.
- To set the magnification to 100%, click the **Zoom to 100%** button in the workspace toolbar.
- To resize the script layout to the size of the workspace, click the **Zoom to Fit Graph in Window** button in the workspace toolbar.

Drilling Down Into or Up From a Group or Block

Use this procedure to access the script workflow for a group or block.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. To drill down, select a group or block in the workspace and then click **Go down into child graph** in the workspace toolbar.
3. To drill up one level, click **Go up to parent graph** in the workspace toolbar.
4. To go to the top level graph, click **Go to root graph** in the workspace toolbar.

Aligning Objects

Use this procedure to align objects vertically or horizontally.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select the objects that you want to align.
3. In the workspace toolbar, click **Align Selected Objects Vertically** or **Align Selected Object Horizontally**.

Defining Panels

You can perform the following tasks:

Defining the Panel Name

Defining the Panel Label

Defining Panel Pre-Actions

Defining Panel Post-Actions

Defining Panel Text

Substituting a Java Bean for a Panel

Defining the Panel Name

Use this procedure to define the name of a panel. The panel name is how the panel is identified in Oracle Scripting Author.

Prerequisites

Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Properties**.
4. In the Name field, type the name of the panel.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Panel Label

Defining Panel Pre-Actions

Defining Panel Post-Actions

Substituting a Java Bean for a Panel

Defining Panel Text

Defining the Panel Label

Use this procedure to define the label for a panel. The panel label is how the panel is labeled in Oracle Scripting Engine.

Prerequisites

Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Properties**.
4. In the Label field, type the label of the panel.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Panel Name

Defining Panel Pre-Actions

Defining Panel Post-Actions

Substituting a Java Bean for a Panel

Defining Panel Text

Substituting a Java Bean for a Panel

Use this procedure to substitute a Java bean for a panel.

Prerequisites

- Insert a panel into the workspace.
- Write the code for the Java bean.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Properties**.
4. In the Properties pane, select **Replace with a Java Bean**.
5. In the Bean Name field, type the full path and name of the Java bean (for example, mybeans.foobean).
6. In the Jar File Name field, type the full path and name of the jar file for the Java bean.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Panel Name

Defining the Panel Label

Defining Panel Pre-Actions

Defining Panel Post-Actions

Defining Panel Text

Defining Panel Text

You can perform the following tasks:

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Text Editor

References

Defining the Panel Name

Defining the Panel Label

Defining Panel Pre-Actions

Defining Panel Post-Actions

Substituting a Java Bean for a Panel

Entering Panel Text

Use this procedure to define the text that displays in the script panel at runtime.

Prerequisites

Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a panel.
3. From the menu, choose **Tools > Panel Text Editor**.

   The panel text editor appears.
4. Type the panel text.
5. Optionally, select text and then select a format from the **Font** or **Lists** menu.
6. If you want to save the text to the panel, then choose **File > Save** from the menu.
7. If you want to save the text to an HTML file, then choose **File > Export** from the menu.

Note: Exporting the text to an HTML file does not save the text to the panel.

References

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Text Editor

Inserting a Hyperlink

Use this procedure to insert a hyperlink into the text of a script panel.

Prerequisites

- Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a panel.
3. From the menu, choose **Tools > Panel Text Editor**.

   The panel text editor appears.
4. Select or type the text that represents the hyperlink and then click **Insert Link** in the toolbar.

   The selected text is blue and underlined.
5. With the cursor in the text or with the text selected, click **Modify Properties** in the toolbar.

   The Hyperlink dialog box appears.
6. In the hyperlink window, do one of the following:
   - If you want to define the hyperlink using a URL (web address), then type the URL for the hyperlink (for example, http://www.oracle.com) in the URL field.
   - If you want to define the hyperlink using a command, then select Command as Hyperlink and **Edit Command**. (See Defining Commands.)
7. Click **OK** to exit the Hyperlink dialog box.
8. If you want to save the text to the panel, then choose **File > Save** from the menu.
9. If you want to save the text to an HTML file, then choose **File > Export** from the menu.

Note: Exporting the text to an HTML file does not save the text to the panel.

References

Entering Panel Text

Inserting an Embedded Value

Inserting an Image

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Text Editor

Inserting an Embedded Value

Use this procedure to embed a value into the text of a script panel.

Prerequisites

- Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a panel.
3. From the menu, choose **Tools > Panel Text Editor**.

   The panel text editor appears.
4. Type the text placeholder for the embedded value and then click **Insert Embedded Value** in the toolbar.

   The background of the selected text is yellow.
5. With the cursor in the text or with the text selected, click **Modify Properties** in the toolbar.

   The Command dialog box appears.
6. Define a command that returns a value to be displayed in the script panel at runtime. (See Defining Commands.)

   In the panel text editor, the text selected to represent the embedded value now displays the command name.

7. If you want to save the text to the panel, then choose **File > Save** from the menu.
8. If you want to save the text to an HTML file, then choose **File > Export** from the menu.

> **Note:** Exporting the text to an HTML file does not save the text to the panel.

References

Entering Panel Text

Inserting a Hyperlink

Inserting an Image

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Text Editor

Inserting an Image

Use this procedure to insert an image into the text of a script panel.

Prerequisites

- Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a panel.
3. From the menu, choose **Tools > Panel Text Editor**.

   The panel text editor appears.
4. Position the insertion point where you want to insert the picture.
5. In the toolbar, click **Insert Image**.

   The Open dialog box appears.
6. Locate the file that contains the image you want to insert.
7. Click the file and then click **Open**.

The image appears at the insertion point.

8. If you want to save the text to the panel, then choose **File > Save** from the menu.

9. If you want to save the text to an HTML file, then choose **File > Export** from the menu.

Note: Exporting the text to an HTML file does not save the text to the panel.

References

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Text Editor

Exporting Panel Text to an HTML File

Use this procedure to save panel text as an HTML file.

Note: Exporting the text to an HTML file does not save the text to the panel.

Prerequisites

- Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.

2. In the workspace, select a panel.

3. From the menu, choose **Tools > Panel Text Editor**.

   The panel text editor appears.

4. If you want to save the text as an HTML file using its current name and location, then choose **File > Export**.

5. If you want to save the text as an HTML file using a different name and location, then choose **File > Export As**.

   If you choose the Export As command, or if the script has never been exported, then the Save dialog box appears.

6. Specify the location and filename.
7. Click **Save**.

References

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Importing an HTML File into the Panel Text Editor

Importing an HTML File into the Panel Text Editor

Use this procedure to import HTML into the panel text editor.

> **Note:** Importing an HTML file into the panel text editor overwrites any text in the panel text editor.

Prerequisites

- Insert a panel into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, select a panel.
3. From the menu, choose **Tools > Panel Text Editor**.

   The panel text editor appears.

4. Choose **File > Import**.

   The Open dialog box appears.

5. Locate the file that you want to import.

6. Click the file and then click **Open**.

   The HTML appears in the panel text editor.

References

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Exporting Panel Text to an HTML File

Defining Groups

You can perform the following tasks:

Inserting a Group

Importing a Saved Script as a Group

Defining the Group Name

Defining Group Pre-Actions

Defining Group Post-Actions

Defining Group Permission

Importing a Saved Script as Group

Use this procedure to import a saved script workflow into the current script project. The script workflow is imported as a group.

Prerequisites

None

Steps

1. Choose **File** > **Import**.

   The Open dialog box appears.

2. Locate the script that you want to import.
3. Click the file and then click **OK**.

   The script appears in the workspace as a group.

References

Inserting a Group

Defining the Group Name

Defining Group Pre-Actions

Defining Group Post-Actions

Defining Group Permission

Defining the Group Name

Use this procedure to define the name of a group. The group name is how the group is identified in Oracle Scripting Author.

Prerequisites

Insert a group into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a group.

   The Properties dialog box appears.
3. In the Group tree, select **Properties**.
4. In the Name field, type the name of the group.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Group Pre-Actions

Defining Group Post-Actions

Defining Group Permission

Defining Group Permissions

Use this procedure to restrict access to the script workflow for a group at runtime.

Prerequisites

Insert a group into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a group.

   The Properties dialog box appears.
3. In the Group tree, select **Permissions**.

4. If you want all users to be able to access the script workflow for the group, then select **All can access**.
5. If you want no users to be able to access the script workflow for the group, then select **None can access**.
6. If you want specific users to be able to access the script workflow for the group, then do the following:
   a. Select **Define allowed access**.
   b. Click **Add**.

      The Permission Entry dialog appears.
   c. In the Name field, type the user login.
   d. In the ID field, type the user ID.
   e. Click OK to exit the Permission Entry dialog box.
   f. If you want to add another user, then repeat steps b through e.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Group Name

Defining Group Pre-Actions

Defining Group Post-Actions

Defining the Script Workflow

You can perform the following tasks:

Defining Panel Answer Controls

Defining Branches

Defining Actions

Integrating with Data Sources

Defining Commands

Defining Panel Answer Controls

You can perform the following tasks:

Substituting a Java Bean for an Answer

Defining a Button in a Script Panel

Defining a Text Field in a Script Panel

Defining a Multi Line Text Area in a Script Panel

Defining a Drop Down List in a Script Panel

Defining a Radio Button Group in a Script

Defining a Check Box Group in a Script Panel

Defining the Answer Control Layout in a Script Panel

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

Use this procedure to substitute a Java bean for a panel.

Prerequisites

Write the code for the Java bean.

Insert a panel into the script workspace.

Define a panel answer

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, select **Add**.

   The Answer Entry Dialog dialog box appears.
5. In the Java Bean area, do the following:

   a. In the Bean Name field, type the full path and name of the Java bean (for example, mybean.answerbean).

   b. In the Jar Name field, type the full path and name of the jar file for the Java bean.
6. Click **OK** to exit the Answer Entry Dialog dialog box.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining a Button in a Script Panel

Defining a Text Field in a Script Panel

Defining a Multi Line Text Area in a Script Panel

Defining a Drop Down List in a Script Panel

Defining a Radio Button Group in a Script

Defining a Check Box Group in a Script Panel

Defining the Answer Control Layout in a Script Panel

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Defining a Button in a Script Panel

Use this procedure to define one or more buttons in a script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, click **Add**.

   The Answer Entry Dialog dialog box appears.
5. If this is the only answer control for this panel or if this answer control is used for distinct branching, then select **Default for Distinct Branching**.

   Note: One answer control must be set as the trigger for distinct branching — even if distinct branching is not being used by the panel and even if there is only one answer control for the panel.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, choose **Button**.
8. Click **OK** to exit the Answer Entry Dialog dialog box.

The answer appears in the Answer pane in the Properties dialog box.

9. In the Answers pane, select the answer and then click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog box appears.

10. In the Lookups tab, select **Specify lookups** and then click **Add**.

    The Lookup Entry dialog box appears.

11. In the Display Value field, type the text that appears on the button in the script panel at runtime.
12. In the Value field, type the value that is stored in the Oracle Scripting database schema when the button is selected at runtime.
13. Click **OK** to exit the Lookup Entry window.
14. Click **OK** to exit the Edit Data Dictionary dialog box.
15. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining a Text Field in a Script Panel

Use this procedure to define one or more text fields in a script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.

3. In the Panel tree, select **Answers**.
4. In the Answers pane, click **Add**.

   The Answer Entry Dialog dialog box appears.

5. If this is the only answer control for this panel or if this answer control is used for distinct branching, then select **Default for Distinct Branching**.

Note: One answer control must be set as the trigger for distinct branching — even if distinct branching is not being used by the panel and even if there is only one answer control for the panel.

---

6. In the Name field, type the name of the answer.

Note: The answer name must be unique.

---

7. From the UI Type list, select **Text Field**.
8. In the UI Label field, type the label that appears next to the text field in the script panel at runtime.
9. Click **OK** to exit the Answer Entry Dialog dialog box.

   The answer appears in the Answer pane in the Properties dialog box.
10. If you want to define another text field in the same script panel, then repeat steps 4 through 9.
11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining a Multi Line Text Area in a Script Panel

Use this procedure to define one or more multi line text areas in a script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, click **Add**.

   The Answer Entry Dialog dialog box appears.

5. If this is the only answer control for this panel or if this answer control is used for distinct branching, then select **Default for Distinct Branching**.

> **Note:** One answer control must be set as the trigger for distinct branching — even if distinct branching is not being used by the panel and even if there is only one answer control for the panel.

6. In the Name field, type the name of the answer.

> **Note:** The answer name must be unique.

7. From the UI Type list, select **Text Area**.
8. In the UI Label field, type the label that appears next to the text area in the script panel at runtime.
9. Click **OK** to exit the Answer Entry Dialog dialog box.

   The answer appears in the Answer pane in the Properties dialog box.
10. If you want to define another text area in the same script panel, then repeat steps 4 through 9.
11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining a Drop Down List in a Script Panel

Use this procedure to define one or more drop down lists in a script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.

4. In the Answers pane, click **Add**.

   The Answer Entry Dialog dialog box appears.

5. If this is the only answer control for this panel or if this answer control is used for distinct branching, then select **Default for Distinct Branching**.

   Note: One answer control must be set as the trigger for distinct branching — even if distinct branching is not being used by the panel and even if there is only one answer control for the panel.

   ---

6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.

   ---

7. From the UI Type list, choose **Drop Down**.
8. Optionally, in the UI Label field, type the label that appears next to the drop down list in the script panel at runtime.
9. Click **OK** to exit the Answer Entry Dialog dialog box.

   The answer appears in the Answer pane in the Properties dialog box.

10. In the Answers pane, select the answer and then click **Edit Data Dictionary**.

    The Edit Data Dictionary dialog box appears.

11. In the Lookups tab, select **Specify lookups** and then click **Add**.

    The Lookup Entry dialog box appears.

12. In the Display Value field, type the text that appears in the drop down list in the script panel at runtime.
13. In the Value field, type the value that is stored in the Oracle Scripting database schema when the drop down is selected at runtime.
14. Click **OK** to exit the Lookup Entry window.
15. If you want to add another item to the drop down list, then repeat steps 11 through 14.
16. Click **OK** to exit the Edit Data Dictionary dialog box.

17. If you want to define another drop down list in the same script panel, then repeat steps 4 through 16.
18. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining a Radio Button Group in a Script

Use this procedure to define one or more radio button groups in a script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, click **Add**.

   The Answer Entry Dialog dialog box appears.
5. If this is the only answer control for this panel or if this answer control is used for distinct branching, then select **Default for Distinct Branching**.

   Note: One answer control must be set as the trigger for distinct branching — even if distinct branching is not being used by the panel and even if there is only one answer control for the panel.

6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.

7. From the UI Type list, choose **Radio Button**.
8. Optionally, in the UI Label field, type the label that appears next to the radio button group in the script panel at runtime.

9. Click **OK** to exit the Answer Entry Dialog dialog box.

   The answer appears in the Answer pane in the Properties dialog box.

10. In the Answers pane, select the answer and then click **Edit Data Dictionary**.

    The Edit Data Dictionary dialog box appears.

11. In the Lookups tab, select **Specify lookups** and then click **Add**.

    The Lookup Entry dialog box appears.

12. In the Display Value field, type the text that appears next to the radio button in the script panel at runtime.

13. In the Value field, type the value that is stored in the Oracle Scripting database schema when the radio button is selected at runtime.

14. Click **OK** to exit the Lookup Entry window.

15. If you want to add another radio button to the group, then repeat steps 11 through 14.

16. Click **OK** to exit the Edit Data Dictionary dialog box.

17. If you want to define another radio button group in the same script panel, then repeat steps 4 through 16.

18. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining a Check Box Group in a Script Panel

Use this procedure to define a one or more check boxes in a script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.

2. In the workspace, double-click a panel.

   The Properties dialog box appears.

3. In the Panel tree, select **Answers**.

4. In the Answers pane, click **Add**.

The Answer Entry Dialog dialog box appears.

5. If this is the only answer control for this panel or if this answer control is used for distinct branching, then select **Default for Distinct Branching**.

Note: One answer control must be set as the trigger for distinct branching — even if distinct branching is not being used by the panel and even if there is only one answer control for the panel.

6. In the Name field, type the name of the answer.

Note: The answer name must be unique.

7. From the UI Type list, choose **Check Box**.
8. In the UI Label field, type the label that appears next to the check box in the script panel at runtime.
9. Click **OK** to exit the Answer Entry Dialog dialog box.

   The answer appears in the Answer pane in the Properties dialog box.
10. If you want to define another check box in the same script panel, then repeat steps 4 through 9.
11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining the Answer Control Layout in a Script Panel

Use this procedure to choose the layout of the answer controls in the script panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.

3. In the Panel tree, select **Properties**.
4. In the Answers pane, do one of the following:
   - If you want to vertically align all answer controls for the script panel, then select **Vertical Answer Layout**.
   - If you want to horizontally align all answer controls for the script panel, then select **Horizontal Answer Layout**.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Data Constraints for an Answer Control

Use this procedure to define data constraints for an answer control.

Prerequisites

Insert a panel into the script workspace.

Define an answer control for the script panel.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, select the answer and then click **Edit Data Dictionary**.

   The Data Dictionary dialog box appears.
5. Define the data constraints for the answer control.
6. Click **OK** to exit the Data Dictionary dialog box.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Reordering Answer Options

Use this procedure to order the options in an answer control that has a list of values (such as a radio button group or check box group).

Prerequisites

Insert a panel into the script workspace.

Define an answer control for the script panel.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, select the answer and then click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog box appears.
5. In the Lookups tab, select a value.
6. If you want to move the option up, then click **Move Up**.
7. If you want to move the option down, then click **Move Down**.
8. Click **OK** to exit the Edit Data Dictionary dialog box.
9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Reordering Answer Controls

Use this procedure to order the answer controls in a script panel.

Prerequisites

Insert a panel into the script workspace.

Define an answer control for the script panel.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.

4. In the Answers pane, select an answer.
5. If you want to move the answer control up, then click **Move Up**.
6. If you want to move the answer control down, then click **Move Down**.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Deleting Answer Options from an Answer Control

Use this procedure to delete an options from an answer control.

Prerequisites

Insert a panel into the script workspace.

Define an answer control for the script panel.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, select an answer and then click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog box appears.
5. In the Lookups tab, select a value and then click **Remove**.
6. Click **OK** to exit the Edit Data Dictionary dialog box.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Deleting Answer Controls from a Script Panel

Use this procedure to delete an answer controls from a script panel.

Prerequisites

Insert a panel into the script workspace.

Define an answer control for the script panel.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. In the Answers pane, select an answer and then click **Remove**.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Branches

You can perform the following tasks:

Defining a Default Branch

Defining a Distinct Branch

Defining a Conditional Branch

Defining an Indeterminate Branch

Defining the Branch Name

Reordering Branches

Defining a Default Branch

Use this procedure to navigate the script to a default destination. This type of branch is also used when there is only one destination.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Default Branch Mode** tool.

   When the pointer is over the workspace, the pointer is a cross hair (+).
2. In the workspace, drag the pointer from the source object to the destination object.

When you release the pointer over the destination object, the branch arrow appears. The branch is red. This indicates that the branch is currently selected.

Defining a Distinct Branch

Use this procedure to navigate the script to a object based on an answer selected in the preceding panel.

Prerequisites

Insert a panel into the script workspace.

Insert a distinct branch.

Define an answer control for the script panel.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click an indeterminate branch.

   The Properties dialog box appears.
3. In the Distinct tree, select **Value**.

   In the Value pane, the Current Unused Answers list contains the display values for the default answer defined in the source panel that have not been assigned to a distinct branch.
4. In the Value pane, do the following:

   a. In the Current Unused Answers list, click the answer that triggers the branch.

   b. Click the double right-arrow button to move the answer to the Selected Answer list.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining a Conditional Branch

Use this procedure to navigate the script to a object based on the evaluation of a conditional expression as true.

Prerequisites

Insert a panel into the script workspace.

Insert a conditional branch.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click an indeterminate branch.

   The Properties dialog box appears.
3. In the Conditional tree, select **Condition**.
4. In the Condition pane, click **Edit**.

   The Command dialog box appears.
5. Define the condition as a command.

   The command must return true or false.
6. Click **OK** to exit the Command dialog box.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining an Indeterminate Branch

Use this procedure to navigate the script based on the evaluation of an expression that returns the name of the destination object.

Prerequisites

Insert an indeterminate branch.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click an indeterminate branch.

   The Properties dialog box appears.
3. In the Indeterminate tree, select **Expression**.
4. In the Expression pane, click **Edit**.

   The Command dialog box appears.

5. Define the indeterminate expression as a command.

   The command must return the name of a panel, block, or group that is on the same graph as the source object.

6. Click **OK** to exit the Command dialog box.

7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining the Branch Name

Use this procedure to define the name of the branch. Except for default branches, the branch name appears with the branch in the script workspace.

Prerequisites

Insert a branch into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.

2. In the workspace, double-click a branch.

   The Properties dialog box appears.

3. In the Panel tree, select **Properties**.

4. In the Name field, type the name of the branch.

5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

6. Optionally, double-click a branch label in the script workspace and then type the branch name.

Reordering Branches

Use this procedure to place branches in the order in which they are to be evaluated.

Prerequisites

Insert a panel into the script workspace.

Insert branches into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Branches**.
4. In the Branches pane, select a branch.
5. If you want to evaluate the branch earlier, then click **Move Up**.
6. If you want to evaluate the branch later, then click **Move Down**.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Actions

You can perform the following tasks:

Defining Script Pre- and Post-Actions

Defining Panel Pre- and Post-Actions

Defining Group Pre- and Post-Actions

Defining Block Pre- and Post-Actions

Defining Branch Actions

Defining Script Pre- and Post-Actions

Use this procedure to define an action to take place before or after a script.

Prerequisites

Create or open a script.

Steps

1. Choose **File > Script Properties** or double-click in the workspace away from any objects.

   The Properties dialog box appears.
2. In the Script tree, expand **Actions** and then select **Pre Actions** or **Post Actions**.

3. In the Actions pane, click **Add**.

   The Command dialog box appears.

4. Define a command for the action. (See Defining Commands.)
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Panel Pre- and Post-Actions

Use this procedure to define an action to take place before or after a panel.

Prerequisites

Insert a panel into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a panel.

   The Properties dialog box appears.

3. In the Panel tree, expand **Actions** and then select **Pre Actions** or **Post Actions**.
4. In the Actions pane, click **Add**.

   The Command dialog box appears.

5. Define a command for the action. (See Defining Commands.)
6. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Group Pre- and Post-Actions

Use this procedure to define an action to take place before or after a group.

Prerequisites

Insert a group into the workspace

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.

2. In the workspace, double-click a group.

   The Properties dialog box appears.
3. In the Group tree, expand **Actions** and then select **Pre Actions** or **Post Actions**.
4. In the Actions pane, click **Add**.

   The Command dialog box appears.
5. Define a command for the action. (See Defining Commands.)
6. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Block Pre- and Post- Actions

Use this procedure to define an action to take place before or after a block.

Prerequisites

Insert a block into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, expand **Actions** and then select **Pre Actions** or **Post Actions**.
4. In the Actions pane, click **Add**.

   The Command dialog box appears.
5. Define a command for the action. (See Defining Commands.)
6. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Branch Actions

Use this procedure to define an action to take place upon branching

Prerequisites

Insert branches into the script workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a branch.

   The Properties dialog box appears.
3. In the tree, select **Actions**.
4. In the Actions pane, click **Add**.

   The Command dialog box appears.
5. Define a command for the action. (See Defining Commands.)
6. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Integrating with Data Sources

You can perform the following tasks:

Inserting a Block

Defining the Block Name

Defining Block Pre-Actions

Defining Block Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Defining a Database Update Block

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Defining the Block Name

Use this procedure to define the name of a block. The block name is how the block is identified in Oracle Scripting Author.

Prerequisites

Insert a block into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Properties**.
4. In the Name field, type the name of the block.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Block Pre-Actions

Defining Block Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Defining a Database Update Block

Defining a Database Query Block

Use this procedure to define an Oracle Scripting business object that searches data in a database and retrieves the records that match your search criteria.

Prerequisites

Insert a block into the workspace. In addition, you need the following information:

- The names of the database tables that contain the information that you want to search or retrieve.

- The names of the table columns that contain the search criteria and that contain the information that you want to retrieve.
- The names of the primary keys and, if any, the foreign keys of the tables.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Type**.
4. In the Type pane, select **Query Block**.
5. In the Block tree, select **Object dictionary**.
6. In the Connection / Table tab, do the following:
   a. Define how Oracle Scripting connects to the database at runtime. (See Defining Database Connections for a Block.)

   b. List the database tables that contain the information that you want to search or retrieve. (See Defining Database Tables for a Block.)
7. If you want to query data from more than one database table, then, in the Join Condition tab, indicate the relationship between related tables. (See Defining Join Conditions for a Block.)
8. In the Query Constraints tab,
   a. If you want to improve network performance during the query, then type the number of rows to be returned at a time during retrieval of the query result in the Prefetch Value field.

   Note: The row prefetch value is a feature of Oracle JDBC drivers. Standard JDBC drivers return the query result one row at a time.

   b. Optionally, in the Primary Table field, type the name of the primary table.

   c. If the rows retrieved by the query are to be locked for a subsequent update, then select **Query for update**.

   d. Define your search criteria. (See Defining Query Constraints for a Query Block.)

   e. List the information that you want to retrieve. (See Defining Query Columns for a Query Block.)

9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Block Name

Defining Block Pre-Actions

Defining Block Post-Actions

Defining a Database Insert Block

Defining a Database Update Block

Defining a Database Insert Block

Use this procedure to define an Oracle Scripting business object that inserts a record into a database.

Prerequisites

Insert a block into the workspace. In addition, you need the following information:

- The names of the database tables into which you want to insert a record.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Type**.
4. In the Type pane, select **Insert Block**.
5. In the Block tree, select **Object dictionary**.
6. In the Connection/Table tab, do the following:

   a. Define how Oracle Scripting connects to the database at runtime. (See Defining Database Connections for a Block.)

   b. List the database tables into which you want to insert information. (See Defining Database Tables for a Block.)

7. If you want to insert data into more than one database table, then, in the Join Condition tab, indicate the relationship between related tables. (See Defining Join Conditions for a Block.)

   Note: If there is only one table in the block or if the tables in the block are not related, then you must still indicate the primary keys for each table.

   ---

8. If you want to insert a panel answer into a database table, then do the following:

   a. Drill down into the block

   b. Insert a panel in the block.

   c. Insert a termination point into the script workspace.

   a. Draw branches between the objects to indicate the flow of the script.

   b. Define an answer. (See Defining Panel Answer Controls.)

   c. In the Answers pane, select the answer and then click **Data Dictionary**.

      The Data Dictionary dialog appears. The default data dictionary name is *answername*DataDict.

   d. In the Table field, type the name of the table into which the answer is inserted.

   e. In the Column field, type the name of the column into which the answer is inserted.

   f. Click **OK** to exit the Data Dictionary dialog box.

   Note: Answers must be listed according to the default order of the columns in the table.

   ---

   g. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

9. If you want to insert a panel answer into an additional table or to insert a value that is not represented by a panel answer into a table, then do the following:

   a. In the Data Constraints tab of the insert block, click **Add**.

The Data Constraint dialog appears.

b. In the Command area, click Edit.

The Command dialog box appears.

c. Define a command that obtains or derives the value to be inserted. (See Defining Commands.)

d. In the Name field, type the name of the data constraint.

Note: After the command is executed, the value is stored on the Oracle Scripting blackboard under the name of the data constraint.

e. In the Table field, type the name of the table into which the value is inserted.

f. In the Column field, type the name of the column into which the value is inserted.

g. Click **OK** to exit the Data Constraints dialog box.

h. If you want to add another data constraint, then repeat steps a through g.

10. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Block Name

Defining Block Pre-Actions

Defining Block Post-Actions

Defining a Database Query Block

Defining a Database Update Block

Defining a Database Update Block

Use this procedure to define an Oracle Scripting business object that updates a record in a database.

Prerequisites

Insert a block into the workspace. In addition, you need the following information:

- The names of the database tables that you want to update.

Note: An update block must be preceded by a query block that retrieves at least one valid row.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Type**.
4. In the Type pane, select **Update Block**.
5. In the Block tree, select **Object dictionary**.
6. In the Connection / Table tab, do the following:

   a. Define how Oracle Scripting connects to the database at runtime. (See Defining Database Connections for a Block.)

   b. List the database tables that contain the information that you want to update. (See Defining Database Tables for a Block.)
7. If you want to update data in more than one database table, then, in the Join Condition tab, indicate the relationship between related tables. (See Defining Join Conditions for a Block.)

   Note: If there is only one table in the block or if the tables in the block are not related, then you must still indicate the primary keys for each table.

   ---
8. If you want to use a panel answer to update a database table, then do the following:

   a. Drill down into the block

   b. Insert a panel in the block.

   c. Insert a termination point into the script workspace.

   d. Draw branches between the objects to indicate the flow of the script.

e. Define an answer. (See Defining Panel Answer Controls.)

f. In the Answers pane, select the answer and then click **Data Dictionary**.

   The Data Dictionary dialog appears. The default data dictionary name is *answername*DataDict.

g. In the Table field, type the name of the table to be updated.

h. In the Column field, type the name of the column to be updated.

i. Click **OK** to exit the Data Dictionary dialog box.

Note: Answers must be listed according to the default order of the columns in the table.

---

j. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

9. If you want to update an additional table with a panel answer or to update a table with a value that is not represented by a panel answer, then do the following:

   a. In the Data Constraints tab of the update block, click **Add**.

      The Data Constraint dialog appears.

   b. In the Command area, click Edit.

      The Command dialog box appears.

   c. Define a command that obtains or derives the value used to update the table. (See Defining Commands.)

   d. In the Name field, type the name of the data constraint.

Note: After the command is executed, the value is stored on the Oracle Scripting blackboard under the name of the data constraint.

---

   e. In the Table field, type the name of the table to be updated.

   f. In the Column field, type the name of the column to be updated.

   g. Click **OK** to exit the Data Constraints dialog box.

   h. If you want to add another data constraint, then repeat steps a through g.

10. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining the Block Name

Defining Block Pre-Actions

Defining Block Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Defining Database Connections for a Block

Use this procedure to define how Oracle Scripting connects to the database at runtime prior to the execution of the block.

Prerequisites

Insert a block into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Object dictionary**.
4. If you want to define new connection parameters, then, in the Connection / Tables tab, do the following:

   a. If necessary, clear the **Reuse an existing connection** box.

   b. Type the connection parameters. (See Guidelines.)

   Note: When you save your work in the Properties dialog box, the connection parameters are available in the Existing Connections list for this script and can be reused in another block or when deploying the script.

   ---

5. If you want to reuse an existing connection, then, in the Connections/Tables tab, do the following:
   a. Select the **Reuse an existing connection** box.
   b. From the Existing Connections list, select a connection.
6. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Guidelines

Connection JDBC Driver. Specifies the Java class to be used as the driver for connecting to the Oracle database. Only the thin Oracle JDBC driver is supported. Use the following string:

```
oracle.jdbc.driver.OracleDriver
```

Connection URL. Specifies the database instance for the connection. Use the following format:

```
jdbc:oracle:thin:@<hostname>:<TNS port>:<SID>
```

The `<hostname>` parameter specifies the name of the database server machine. The `<TNS port>` parameter specifies the port number for TNS connections. This value is determined by the configuration of the database. The `<SID>` parameter specifies the SID of the database or the instance name.

User Name Administrator Oracle8*i* database login

Password Administrator password

References

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Defining Database Tables for a Block

Use this procedure to list the tables that you want to query, update, or insert.

Prerequisites

Insert a block into the workspace. In addition, you need the names of the database tables that contain the information that you want to query, update, or insert.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Object dictionary**.
4. In the Connections / Table tab in the Tables area, click **Add table**.

   The Table dialog box appears.
5. Type the name of the table and then click **OK** to exit the Table dialog box.
6. If you want to add another table, then repeat steps 4 through 5.

   Note: List master tables first and then tables related by foreign key.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Database Connections for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Defining Join Conditions for a Block

Use this procedure to indicate the links between related tables. Tables are joined according to common values existing in corresponding columns.

Note: If there is only one table in the block or if the tables in the block are not related, then you must still indicate the primary keys for each table.

The primary key is a column or set of columns that uniquely identifies each row of a table. The foreign key is a column or set of columns that references a primary or unique key in the same or a different table.

Prerequisites

Insert a block into the workspace. In addition, you need the names of the primary keys and foreign keys, if any, of the tables that you want to query, update, or insert.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a block.

   The Properties dialog box appears.
3. In the Block tree, select **Object dictionary**.
4. In the Join Condition tab, click **Add**.

   The Join Condition Dialog dialog box appears.
5. In the Master PK Table tab, do the following:

   a. In the Master Table field, type the name of the table.

   b. Click **Add Master PK**.

   The Column Entry dialog box appears.

   Note: Prefix the column name with the table name (for example, *table1.column1*).

   c. Type the name of the column that is the primary key of the master table and then click **OK** to exit the Column Entry dialog box.

   d. If you want to add another primary key, then repeat steps b through c.

6. In the Detail PK Table tab, do the following:

   a. In the Detail Table field, type the name of the related table.

   b. Click **Add Detail PK**.

   The Column Entry dialog box appears.

   > **Note:** Prefix the column name with the table name (for example, *table1.column1*).

   c. Type the name of the column that is the primary key of the related table and then click **OK** to exit the Column Entry dialog box.

   d. If you want to add another primary key, then repeat steps b through c.

7. In the Detail FK Table tab, do the following:

   a. Click **Add Detail FK**.

   The Column Entry dialog box appears.

   b. Type the name of the column that links the tables and then click **OK** to exit the Column Entry dialog box.

   c. If you want to add another column name, then repeat steps a through b.

8. Click **OK** to exit the Join Condition Dialog dialog box.

9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Defining Data Constraints for an Insert or Update Block

Use this procedure to list the columns to be inserted or updated.

Prerequisites

Insert a block into the workspace. In addition, you need the names of the database tables that contain the information that you want to update, or insert.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click an insert or update block.

   The Properties dialog box appears.
3. In the Block tree, select **Object dictionary**.
4. In the Data Constraints tab, click **Add**.

   The Data Constraints dialog box appears.
5. Define the data constraints.
6. Click **OK** to exit the Data Constraints dialog box.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Defining Query Constraints for a Query Block

Use this procedure to define your search criteria for the database query.

Prerequisites

Insert a block into the workspace. In addition, you need the names of the tables and table columns that contain the information that you want to query.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a query block.

   The Properties dialog box appears.
3. In the Block tree, select **Object Dictionary**.
4. In the Query Data tab, in the Query Constraints tab, click Add Constraint.

   The Query Constraint Input dialog box appears.
5. Type the query constraint (for example, *table1.column1 operator value*).

   Note: Conditions based on user input require a panel in the block. The *value* is an answer name in the panel. See Defining Panels within a Block.
6. Click **OK** to exit the Query Constraint Input dialog box.
7. If you want to add another query constraint, then repeat steps 4 through 6.

   Note: Insert logical operators (such as, `NOT`, `AND`, and `OR`) at the end of the query constraint. If no logical operator is used, then `AND` is assumed. Use parentheses at the beginning or end of a query constraint to override the rules of precedence.
8. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Columns for a Query Block

Defining Panels within a Block

Defining Query Columns for a Query Block

Use this procedure to list the information that you want to retrieve from the database.

Prerequisites

Insert a block into the workspace. In addition, you need the names of the tables and table columns that contain the information that you want to query.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a query block.

   The Properties dialog box appears.
3. In the Block tree, select **Object Dictionary**.
4. In the Query Data tab, in the Query Columns tab, click Add Columns.

   The Enter a Key / Value dialog box appears.
5. In the Name field, type the column name.

   Note: Prefix the column name with the table name (for example, *table1.column1*).
6. If you want to improve network performance during the query, then select the type of data in the column from the Data Type list.
7. Click **OK** to exit the Enter a Key / Value dialog box.
8. If you want to add another column, then repeat steps 4 through 7.
9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Panels within a Block

Defining Panels within a Block

Use this procedure to add a panel to a block.

Prerequisites

Insert a block into the workspace.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, click a block and then click the **Go down into child graph button** in the workspace toolbar.

   The graph for the block appears.
3. Define the graph.

References

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Commands

You can perform the following tasks:

Invoking a Public Method in a Java Class

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Invoking a Public Method Exposed by a COM Automation Server

Setting a Constant

Retrieving a Panel Answer

Invoking a Public Method in a Java Class

Use this procedure to invoke a public method in a Java class and return the value returned by the invocation, if any. The Java command object takes zero or more parameters and returns zero or one value.

Prerequisites

Write the code that handles the data exchange.

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **Java Command**.
3. In the Command Info area, do the following:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the command string as follows:
      ```
      <fully qualified class name>::<method name>
      ```
4. If you want to list parameters for the command object, then, in the Parameters area, do the following:
   a. Click **Add**.

      The Parameters dialog box appears.

   b. In the Name field, type the name of the parameter.
   c. In the Value field, type the literal string value for the parameter.

   d. If you want to add the value as the return value of an executed command, then select **Add Value as Command** and then click **Edit** in the Value field.

   Note: A command object used as a parameter to another command must return a string value.

   ---

   e. Click **OK** to exit the Parameters dialog box.
   f. If you want to add another parameter, then repeat steps a through e.
5. Click **OK** to exit the Command dialog box.
6. Save your work.

References

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Invoking a Public Method Exposed by a COM Automation Server

Setting a Constant

Retrieving a Panel Answer

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Use this procedure to invoke a PL/SQL stored procedure or function in an Oracle database and return the value returned by the invocation, if any. The PL/SQL command object takes zero or more parameters and returns zero or one value.

Prerequisites

Write the code that handles the data exchange.

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **PL/SQL Command**.
3. In the Command Info area, do the following:
   a. In the Name field, type the name of the command.

   b. In the Command field, type the name of a PL/SQL stored procedure or function in an Oracle database.

4. If you want to list parameters for the command object, then, in the Parameters area, do the following:

   a. Click **Add**.

      The Parameters dialog box appears.

   b. In the Name field, type the name of the parameter.

   c. In the Value field, type the literal string value for the parameter.

   d. If you want to add the value as the return value of an executed command, select **Add Value as Command** and then click **Edit** in the Value field.

      Note: A command object used as a parameter to another command must return a string value.

   e. In the In/Out list, select the parameter type.

   f. Click **OK** to exit the Parameters dialog box.

   g. If you want to add another parameter, then repeat steps a through f.

5. If you want to connect to the Oracle database using new connection parameters, then do the following:

   a. Clear the **Reuse an existing connection** box.

   b. Type the connection parameters. (See Guidelines.)

6. If you want to reuse existing connection parameters to connect to the Oracle database schema, then do the following:

   a. Select the **Reuse an existing connection** box.

   b. From the Existing Connections list, select a connection.

   c. In the Password field, type the password for the selected database connection.

7. Click **OK** to exit the Command dialog box.

8. Save your work.

Guidelines

Connection JDBC Driver. Specifies the Java class to be used as the driver for connecting to the Oracle database. Only the thin Oracle JDBC driver is supported. Use the following string:

```
oracle.jdbc.driver.OracleDriver
```

Connection URL. Specifies the database instance for the connection. Use the following format:

```
jdbc:oracle:thin:@<hostname>:<TNS port>:<SID>
```

The `<hostname>` parameter specifies the name of the database server machine. The `<TNS port>` parameter specifies the port number for TNS connections. This value is determined by the configuration of the database. The `<SID>` parameter specifies the SID of the database or the instance name.

User Name Administrator Oracle8*i* database login

Password Administrator password

References

Invoking a Public Method in a Java Class

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Invoking a Public Method Exposed by a COM Automation Server

Setting a Constant

Retrieving a Panel Answer

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Use this procedure to invoke a PL/SQL procedure or function defined in an Oracle Forms PL/SQL package on the Oracle Forms server and return the value returned by the invocation, if any. The Forms command object takes zero or more parameters and returns zero or one value.

Prerequisites

Write the code that handles the data exchange.

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **Forms Command**.
3. In the Command Info area, do the following:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the command string.
4. If you want to list the parameters for the command object, then, in the Parameters area, do the following:
   a. Click **Add**.

      The Parameters dialog box appears.

   b. In the Name field, type the name of the parameter.
   c. In the Value field, type the literal string value for the parameter.
   d. If you want to add the value as the return value of an executed command, then select **Add Value as Command** and then click **Edit** in the Value field.

      Note: A command object used as a parameters to another command must return a string value.

   e. In the In/Out list, select the parameters type.
   f. Click **OK** to exit the Parameters dialog box.
   g. If you want to add another parameter, then repeat steps a through f.
5. If you want to define a return value, then, in the Return Value area, do the following:
   a. Click **Edit**.

      The Parameters dialog box appears.

   b. In the Name field, type the name of the return value.
   c. Click **OK** to exit the Parameters dialog box.
6. Click **OK** to exit the Command dialog box.
7. Save your work.

References

Invoking a Public Method in a Java Class

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Invoking a Public Method Exposed by a COM Automation Server

Setting a Constant

Retrieving a Panel Answer

Invoking a Public Method Exposed by a COM Automation Server

Use this procedure to invoke a public method exposed by a COM Automation server and return the value returned by the invocation, if any. The COM command object takes zero or more parameters and returns zero or one value.

Prerequisites

Write the code that handles the data exchange.

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **COM Command**.
3. In the Command Info area, do the following:
    a. In the Name field, type the name of the command.
    b. In the Command field, type the command string.
4. If you want to list the parameters for the command object, then, in the Parameters area, do the following:
    a. Click **Add**.

       The Parameters dialog box appears.
    b. In the Name field, type the name of the parameter.
    c. In the Value field, type the literal string value for the parameter.
    d. If you want to add the value as the return value of an executed command, then select **Add Value as Command** and then click **Edit** in the Value field.

Note: A command object used as a parameters to another command must return a string value.

---

e. Click **OK** to exit the Parameters dialog box.

f. If you want to add another parameter, then repeat steps a through e.

5. If you want to define a return value, then, in the Return Value area, do the following:

   a. Click **Edit**.

      The Parameters dialog box appears.

   b. In the Name field, type the name of the return value.

   c. Click **OK** to exit the Parameters dialog box.

6. Click **OK** to exit the Command dialog box.

7. Save your work.

References

Invoking a Public Method in a Java Class

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Setting a Constant

Retrieving a Panel Answer

Setting a Constant

Use this procedure to return a constant value. The Constant command object takes no parameters and returns one value.

Prerequisites

None

Steps

1. Navigate to a Command dialog box.

2. In the Command Type area, select **Constant Command**.

3. In the Command Info area, type the name of the command in the Name field.
4. In the Return Value area, do the following:
   a. Click **Edit**.

      The Parameters dialog box appears.
   b. In the Value field, type the return value.
   c. Click **OK** to exit the Parameters dialog box.
5. Click **OK** to exit the Command dialog box.
6. Save your work.

References

Invoking a Public Method in a Java Class

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Invoking a Public Method Exposed by a COM Automation Server

Retrieving a Panel Answer

Retrieving a Panel Answer

Use this procedure to return a value from the Oracle Scripting blackboard. The Blackboard command object takes no parameters and returns one value.

Prerequisites

None

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **Blackboard Command**.
3. In the Command Info area, do the following:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the name of the panel answer to be returned.

Note: Panel answers are listed in the Properties dialog for a panel, on the Answers pane.

4. Click **OK** to exit the Command dialog box.
5. Save your work.

References

Invoking a Public Method in a Java Class

Invoking a PL/SQL Stored Procedure or Function in an Oracle Database

Invoking a PL/SQL Procedure or Function in an Oracle Forms Server

Invoking a Public Method Exposed by a COM Automation Server

Setting a Constant

Defining Script Attributes

You can perform the following tasks:

Defining the Script Name

Setting Footprinting

Setting Answer Collection for the Script

Defining the Script Name

Use this procedure to define the name of the script.

Prerequisites

None

Steps

1. Choose **File** > **Script Properties** or double-click in the workspace away from any objects.

   The Properties dialog box appears.
2. In the Script tree, select **Properties**.
3. In the Name field, type the name of the script.
4. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Setting Footprinting

Use this procedure to have Oracle Scripting record the start time and end time for every script panel that is enabled during an agent interaction.

Prerequisites

Create or open a script.

Steps

1. Choose **File** > **Script Properties** or double-click in the workspace away from any objects.

   The Properties dialog box appears.

2. In the Script tree, select **Properties**.
3. To record the start time and end time for every script panel that is enabled during an agent interaction, select **Footprinting**.
4. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Setting Answer Collection for the Script

Setting Answer Collection for the Script

Use this procedure to have Oracle Scripting record the answers to the script panels enabled during an agent interaction.

Prerequisites

Create or open a script.

Steps

1. Choose **File > Script Properties** or double-click in the workspace away from any objects.

   The Properties dialog box appears.
2. In the Script tree, select **Properties**.
3. To record the answers to the script panels enabled during an agent interaction, select **Answer Collection**.
4. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

References

Setting Answer Collection for the Script

Defining the Interface Attributes of the Script Engine

You can perform the following tasks:

Defining the Script Header

Defining Script Shortcuts

Programming the Script Disconnect Button

Defining the Script Header

Use this procedure to setup the header of the Oracle Scripting interface.

Prerequisites

Create or open a script.

Steps

1. Choose **File** > **Script Properties**.

   The Properties dialog box appears.

2. In the Script tree, select **Static Info Panel**.
3. In the Static Info Panel pane, click an area in the representation of the script header.
4. If you want to insert text, then click **Text**.
5. If you want to insert a timer, then click **Timer**.
6. In the ID field, type an identifying value.

   The Oracle Scripting API requires this ID.

7. In the Label field, type the label that appears in the header area at runtime.
8. In the Command field, click Edit.

   The Command dialog box appears.

9. Define a command for the header text or timer. (See Defining Commands.)
10. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

Defining Script Shortcuts

Use this procedure to define button in the toolbar of the Oracle Scripting interface that provides a shortcut to a group.

Prerequisites

Insert a group.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a group.

   The Properties dialog box appears.
3. In the Group tree, select **Shortcut**.
4. In the Shortcut pane, type the name of the shortcut.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box for the group.
6. Choose **File > Script Properties**.

   The Properties dialog box appears.
7. In the Script tree, select **Shortcut Panel**.
8. In the Shortcut Panel pane, select **Add**.

   Shortcut Info Entry Dialog dialog box appears.
9. In the ID field, type the shortcut name for the group.
10. In the Label field, type the label that appears on the shortcut button at runtime.
11. In the Tooltip field, type an on screen description of the shortcut button that appears when the user's pointer pauses over the button.
12. Click **Edit** in the Command field.

    The Command dialog box appears.
13. Define a command for the shortcut button. (See Defining Commands.)
14. If you want to make the shortcut enabled in the script interface for the compiled script, then select **Enabled**.
15. Click **OK** to exit the Shortcut Info Entry Dialog dialog box.

16. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box for the script.

References

Programming the Script Disconnect Button

Programming the Script Disconnect Button

Use this procedure to program the Disconnect button in the Oracle Scripting interface.

Prerequisites

Insert a group.

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. In the workspace, double-click a group.

   The Properties dialog box appears.
3. In the Group tree, select **Shortcut**.
4. In the Shortcut pane, type **WrapUpShortcut**.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box for the group.

   When the Disconnect button is selected at runtime, the first panel of this group is displayed.

References

Defining Script Shortcuts

Deploying the Script

You can perform the following tasks:

Checking Script Syntax

Deploying a Script to the Database

Checking the Script Syntax

Use this procedure to check the syntax of a script.

Prerequisites

Create or open a script.

Steps

1. Choose **Tools > Syntax Check**.

   When the syntax check is complete, a message appears in the status bar. If there are errors, then the Syntax tab is displayed.

2. In the Syntax tab, click a node to view the error message (if any).

References

Deploying a Script to the Database

Deploying a Script to the Database

Use this procedure to deploy the compiled script to the Oracle Scripting database schema.

Prerequisites

A database user for administration of Oracle Scripting with the following privileges:

- resources
- javasyspriv
- create public synonym
- drop public synonym
- create session

- create procedure

Create or open a script.

Steps

1. Choose **Tools > Deploy Script**.
2. If there are errors, then the Syntax tab is displayed and the Debug pane appears. (See Checking the Script Syntax.) Otherwise, the Deploy Script To dialog box appears.

   Note: Always use a new connection.

3. Type the parameters for connecting to the Oracle Scripting schema.
4. Click **OK** to deploy the script.

Guidelines

Guidelines

Connection JDBC Driver. Specifies the Java class to be used as the driver for connecting to the Oracle database. Only the thin Oracle JDBC driver is supported. Use the following string:

```
oracle.jdbc.driver.OracleDriver
```

Connection URL. Specifies the database instance for the connection. Use the following format:

```
jdbc:oracle:thin:@<hostname>:<TNS port>:<SID>
```

The `<hostname>` parameter specifies the name of the database server machine. The `<TNS port>` parameter specifies the port number for TNS connections. This value is determined by the configuration of the database. The `<SID>` parameter specifies the SID of the database or the instance name.

User Name Administrator Oracle8*i* database login

Password Administrator password

References

Defining the Script Name

Administering Oracle Scripting Engine

This topic group provides task-based procedures for required for ongoing system maintenance and includes information on administration tools and utilities.

This topic group includes the following topic:

Modifying the Parameters for the Oracle Scripting Engine Client

Use this procedure to modify the parameters for Oracle Scripting Engine on the client. These parameters tell the client how to access a script on the server. The parameters are environment variables on the client system.

Prerequisites

Install Oracle Scripting Engine.

Steps

1. On the client, choose Start > Settings > Control Panel.
2. Double-click System.
3. In the Environment tab, modify the parameters:

Engine Parameter	Environment Variable	Comment
Server Host Name	SCRIPTING_SERVER	
Server IIOP Port	SCRIPTING_PORT	Three-tier mode only
Server TNS Port	SCRIPTING_PORT	Two-tier mode only
Server Database SID	SCRIPTING_SID	
Server JNDI Name	SCRIPTING_JNDI	
Three-Tier Mode	SCRIPTING_THREE_TIER	Set to **true**.
Two-Tier Mode	SCRIPTING_THREE_TIER	Set to **false**.
Scrolling Panel Display	SCRIPTING_DISPLAY_MODE	Set to **true**.

Engine Parameter	Environment Variable	Comment
Single Panel Display	SCRIPTING_DISPLAY_MODE	Set to **false**.
Script Name (Specific)	SCRIPTING_SCRIPT_NAME	
Script Name from Database	SCRIPTING_SCRIPT_NAME	Set to **LoadFromDB** to display a list of scripts.
Script Language	SCRIPTING_SCRIPT_LANG	Ignored if script name is loaded from the database.

Customizing Oracle Scripting

This topic group provides information on using APIs to develop or extend the application, changing the user interface, and adding/removing features and functionality from the application.

Scripting Best Practices Overview

This chapter introduces you to the Scripting Best Practices and provides instructions for using the Best Practices scripts that are included in the latest release of Oracle Scripting. The following topics are covered:

- What are Scripting Best Practices?
- How to use the Best Practices Scripts and related components

What Are Scripting Best Practices?

Best Practices Scripting provides off-the-shelf scripts and script templates that create successful call flows for agents on the telephone. Through the Oracle Best Practice Scripts, enterprises can immediately implement scripts into their production environments, effectively bringing the most successful approaches to customer interaction to the agent.

The Best Practice scripts exist as a library of re-usable modules focusing on specific subsets of the customer interaction. This means that an enterprise can “mix and match” a variety of script components to create their own unique call guides, tailored to meet the enterprise’s specific customer relationship guidelines.

Best Practice Scripts also contain built-in data integration points to key database tables and views for querying and updating customer and account related information. For enterprises already using Oracle ERP or CRM applications, this means they can simply move their scripts into production and use the pre-built integration with their existing systems. For those enterprises who have their own external ERP applications, the Best Practice script templates contain pre-built modules which indicate where the “hooks” to the enterprises’ applications can be put in place.   These “hooks” can be created either by Oracle Consulting, or by the enterprise development group, using standard APIs, data sources, and other methods to communicate among multiple applications.

How To Use The Best Practices Scripts

The Best Practice Scripts are an important part of the call guide developers toolkit. Each Best Practice script file consists of an entire script which is installed as part of the Oracle Scripting library. The script can be compiled and executed in a production environment without any modifications to the script. A script developer can adopt all or portions of a Best Practice script and can easily modify the components to create new scripts to meet the particular enterprise need. The following section explains how to use the Best Practice scripts and related components as building blocks in your script development efforts. Chapter 2 of this guide provides descriptions of each of the best practice scripts. The appendix of this guide contains descriptions of the Java and PL/SQL code that is used to support some of the functions within the Best Practice scripts.

First Steps

The first thing a script developer should do before using any of the Best Practices scripts as building blocks is to test them independently unchanged in their environment. Each script file has been delivered as a fully defined script that should execute as-is in a test or production environment without any problems. Use the following procedure to perform your initial script verification.

Deploying the Best Practice Scripts

Steps

1. Open the Oracle Scripting Author.
2. Open one of the Best Practice script files.
3. Deploy the script to the database. This will perform a syntax check first. If the Deploy Script To: dialog box pops up, then the syntax check was successful, with no errors.

   You will need to know the JDBC Driver, URL, User Name and Password that is correct for your environment.

Testing the Deployed Scripts

1. Execute the Oracle Scripting Engine by logging in to the Oracle Applications as a Scripting user.
2. Pick the script to run from the list of deployed scripts.

3. Verify that the script functions properly in your environment, testing all call flow paths.

   If the script does not execute properly then the environment should be checked to ensure all the necessary components have been installed and implemented. If you are testing a script that uses Java commands, verify that the appropriate Java files have been implemented on the server. If you are testing a script that uses Blocks or PL/SQL commands, verify that the appropriate tables, views and PL/SQL packages have been implemented in the database you are using.

   Repeat the above process with each of the Best Practices scripts that you plan to use. Once the scripts execute correctly as-is, then you can begin to modify the scripts or assemble them together as you desire.

Modifying a Best Practice Script

To use a Best Practice script as a building block in your script development efforts, use the following procedure.

1. Open the Oracle Scripting Author.
2. Open one of the Best Practice script files.
3. Make a copy of the script leaving the original as it is by using the Save As menu option.
4. Now using your copy which is currently active in the Scripting workspace, modify the script properties name. You want this name to be different from the Best Practices script name so when you deploy your new script to the database it will not overwrite the existing Best Practices script that is already deployed.
5. Continue to modify the script making changes as appropriate for your implementation.
6. Remember to save your work!
7. Deploy the script to the database.
8. Test your script.

Mixing and Matching Best Practice Scripts

To combine multiple Best Practice scripts into one script you can use the File Import and Export commands. Use the following procedures to include the parts of the scripts that you want into your new script.

Importing an entire script into another script

1. Open the Oracle Scripting Author.
2. Create a new script workspace.
3. Import one of the Best Practice script files that you want to use as part of your new script. The entire script will be copied into the opened script workspace as a Group.
4. Connect the new group to other script objects using the appropriate branch types. Pay particular attention to the transition into and out of the group to ensure that the call flow remains smooth now that the script is being used in a new context.
5. Continue to modify the new group or keep it as it is to meet the needs of your implementation. These changes will not change the original source script. The changes will only apply to the current opened script.
6. Save your work.
7. Deploy the script to the database.
8. Test your script.

Exporting Part of a Script

1. Open the Oracle Scripting Author.
2. Open an existing script.
3. Select a Group within the script. If the script objects you want to export from this script are not already neatly packaged within in a group, you will need to put them into a group first. You can accomplish this by adding a new group to the script workspace and then copying and pasting the objects you want into the group.
4. Export the Group. The group will be saved as a separate script file that can then be readily used as a script building block.
5. Close the script.

Using the Scripting Best Practice Java Code

Many of the Best Practice scripts use supporting Java commands. If you want to change or add to the Java commands you will need to have some Java programming experience. However if you know that a specific piece of logic already exist as a Java command in one script you can easily reuse it within the same script or in another script without changing the underlying Java code. All you need to do is specify the correct Java command and parameter information in the script.

For example, let’s say you want to designate a text field in one of your script panels as a required field. You would do this using a Validation command in the Data Dictionary definition for an Answer. One of the scripts included in the Best Practices is named ‘BP Validation Utilities’. Within this script there are examples of several common validation routines provided, one of which is a validation for a required field. Take a look at the Data Dictionary for the Answer named ‘RequiredEntry’. The validation entry has been defined as a Java command. The command information specifies the Java class and method name joined with two colons. In this example it is ‘BPUtilities::validateRequiredField’. This method requires two parameters. The AnswerPlaceholder parameter is a means of passing the value that the user enters into the text field into the Java method for evaluation. The ControlLabel parameter value is used in the error message text to indicate the field that is being validated.

To reuse this Java command, just select ‘Java Command’ for the Validation command type. Give the command any name you like. Specify the Command string exactly as you see it in the example. Using the Copy and Paste Editing commands come in handy here. Add the Parameter specifications. The AnswerPlaceholder can be left the same as the example. The ControlLabel parameter should be modified to meet the needs of the text field that is being validated. That is, the value ‘Enter Something’ should be replaced with the name of the text field being validated so the error message will refer the user to the appropriate field that is in error.

That’s all there is to it! To see a listing of the various Java commands that are used by the Best Practices scripts, refer to Appendix A. The Java source code files have been included in the release so they can be copied and modified to meet your needs. The compiled class and jar files have also been included in the release for ease of use if your implementation does not require any modifications.

Using the Scripting Best Practices PL/SQL code

Several of the Best Practice scripts also use supporting PL/SQL commands to execute the PL/SQL APIs that update the database. If you want to change or add to the PL/SQL commands you will need to have some PL/SQL programming experience and a good understanding of the data model. However if you know that certain pieces of data are already being manipulated by some existing PL/SQL commands in one script, you can reuse it within the same script or in another script without changing the underlying PL/SQL code.

The easiest way to reuse the PL/SQL commands is to identify the script objects that contain them and then either copy those objects or import them from another script. To see a listing of the various PL/SQL commands that are used by the Best Practices scripts, refer to Appendix B. The PL/SQL source code files have been included in the release so they can be copied and modified to meet your needs.

Best Practices Script Descriptions

This chapter provides descriptions of each of the Best Practices scripts that comes with Oracle Scripting.

The chapter covers the following scripts:

- Call WrapUp Groups
- Call Back Groups
- Do Not Call Group
- Validation Utilities
- Customer/Prospect Update
- Customer Address Update
- Customer Billing Address Query
- Customer Fulfillment Address Update
- Customer Account Query
- Customer Account Suspension Update

Call WrapUp Groups

Group Overview

The sample Call Wrap Up Group scripts would typically be used to display the last panel or set of panels in a script prior to termination, hence you will want to use it as a means to implement the Script Disconnect button. To program the Script Disconnect button using one of the Call WrapUp Group Best Practice samples, use the following procedure.

1. Import the Call WrapUp Group file (for example BP Call WrapUp Group 1.script) into an opened script workspace. A new group blob will be inserted into your script.
2. In the tool palette, click the Toggle Select Mode tool.
3. In the workspace, double-click the new group. The Properties dialog box appears.
4. In the Group tree, select Shortcut.
5. In the Shortcut pane, type WrapUpShortcut.
6. Click OK to save your work and exit the Properties dialog box for the group.
7. Connect the Call WrapUp Group to the Terminal Node using a Default branch.

   When the Disconnect button is selected at runtime, the first panel of this group is displayed.

Note: Only one of the groups within a script should be defined with the Shortcut string of WrapUpShortcut.

Call WrapUp Group 1

Script File Name

BP Call WrapUp Group 1.script

Brief Description

Call WrapUp Group 1 is one of the three samples provided with this set of Scripting Best Practices that demonstrates a mechanism for capturing the call status. It is the most simple example of the three consisting of only a single panel with a single answer node.

High Level Flow

This group consists of a single panel with only a single answer node for recording the call status. The answer node is defined as a drop down control and comes populated with five typical lookup values specified for the call status:

- Contact Made
- Callback Scheduled
- Wrong Number
- Do Not Call
- User Disconnect

The list of lookup values can be easily modified to meet the needs of your implementation by editing the data dictionary for the answer node.

Java Source Code Used

None

Database Tables/Views Used

None

PL/SQL Code Used

None

Call WrapUp Group 2

Script File Name

BP Call WrapUp Group 2.script

Brief Description

Call WrapUp Group 2 is the second of the three samples provided with this set of Scripting Best Practices that demonstrates a mechanism for capturing the call status. It is similar to the first group but allows for capturing three values (Call Outcome, Call Result and Result Reason) instead of just one value to represent the call status.

High Level Flow

This group consists of a single panel with three answer nodes for recording the call status. All three answer nodes are defined as drop down controls and come populated with typical lookup values specified for the Call Outcome, Call Result and Result Reason of a call status.

The lookup values for the Call Outcome are:

- Contact Made
- Contact Not Available
- Busy
- Wrong Number
- Do Not Call
- User Disconnect

The lookup values for the Call Result are:

- N Sale
- Sale
- Customer Complaint
- Call Back
- Multiple Activities
- Completed Activity
- Incomplete Activity

- None (set as default value)

The lookup values for the Result Reason are:

- No Money
- Already Gave
- Too Expensive
- Out of Work
- Gave at the Office
- Other Reason
- Too Busy
- Special Handling Required
- None (set as default value)

The list of lookup values and the default values can be easily modified to meet the needs of your implementation by editing the data dictionary for each of the answer node.

Java Source Code Used:

None

Database Tables/Views Used

None

PL/SQL Code Used

None

Call WrapUp Group 3

Script File Name

BP Call WrapUp Group 3.script

Brief Description

Call WrapUp Group 3 is the final of the three samples provided with this set of Scripting Best Practices that demonstrates a mechanism for capturing the call status. It is the most robust of the three samples. It is similar to the second group in that it captures three values to represent the call status. It however has built in logic to determine whether it is necessary to prompt for the Call Result or Result Reason information based on the values of the Call Outcome and Call Result selections already made.

High Level Flow

This group consists of three panels each with a single answer node. All three answer nodes are defined as drop down controls and come populated with typical lookup values specified for the Call Outcome, Call Result and Result Reason of a call status.

The lookup values for the Call Outcome are:

- Contact Made
- Contact Not Available
- Busy
- Wrong Number
- Do Not Call
- User Disconnect

If Contact Made is selected for the Call Outcome, then the panel for capturing the Call Result will be displayed. If any other answer is selected for the Call Outcome, the script flow will bypass the Call Result and Result Reason panels and go directly to the Terminal Node of the Call WrapUp group.

The lookup values for the Call Result are:

- No Sale
- Sale
- Customer Complaint

- Call Back
- Multiple Activities
- Completed Activity
- Incomplete Activity
- None

If None is selected for the Call Result, then the panel for capturing the Result Reason will be bypassed and the script flow will go directly to the Terminal Node of the Call WrapUp group. If any other answer is selected for the Call Result, the script flow will continue to the next panel to capture the Result Reason.

The lookup values for the Result Reason are:

- No Money
- Already Gave
- Too Expensive
- Out of Work
- Gave at the Office
- Other Reason
- Too Busy
- Special Handling Required
- None

The list of lookup values can be easily modified to meet the needs of your implementation by editing the data dictionary for each answer node. However care should be taken to ensure that a logical flow of the script is maintained by checking the values selected for each of the distinct branches. Additional distinct branches may need to be added.

Java Source Code Used:

None

Database Tables/Views Used

None

PL/SQL Code Used

None

Call Back Groups

Group Overview

The sample Call Back Group scripts are used to facilitate the scheduling of a call back to a customer or prospect. There are three samples provided with this set of Scripting Best Practices. You may find that you like certain features of one group better than some in another. It is perfectly fine to come up with your own Call Back group by mixing and matching various panels from any of the examples offered here or new ones you have created.

Call Back Group 1

Script File Name

BP CallBack Group 1.script

Brief Description

Call Back Group 1 is one of the three sample groups provided to demonstrates a mechanism for capturing information to schedule a call back. It allows the user to select whether the call back should be scheduled for a specific day of the week or for a specific date. It captures the preferred time of day to schedule the call back using 15 minute intervals. It also allows for capturing any notes that will assist in handling the call back.

High Level Flow

If the user selects on the first panel of this group to schedule the call back for a specific day of the week, the next panel that will be displayed will have only one answer node which is a radio button control for selecting the desired day of the week (Monday, Tuesday, Wednesday, etc.).

If the user selects on the first panel of this group to schedule the call back for a specific date, the next panel that will be displayed will have three answer nodes which are used to select the month, day and year to schedule the call back. Each of these answer nodes is defined as a drop down control populated with appropriate data in the lookup values. The month drop down lists the months of the year. The day drop down lists values from 1 to 31. And the year drop down lists the values of 2000, 2001 and 2002 with 2000 currently being defined as the default. The list of lookup values can be easily modified to remove years from the list, add years to the list, or change the year default value by editing the data dictionary for the answer node. Note that no date validation is done in this sample script.

After the specific day of week or date information has been captured, the time of day panel will be displayed. This panel has three answer nodes which are used to select the preferred time of day, the hour and 15 minute interval. The preferred Time of Day answer is defined as a radio button control with a choice of AM or PM. The Hour answer node is defined as a drop down control and is populated with the values of 01 through 12. The Minute answer node is defined as a drop down control and is populated with the values of 00, 15, 30 and 45.

Finally, the script flows to the last panel in the group to capture call back notes in a text area control.

Java Source Code Used

- none

Database Tables/Views Used

- none

PL/SQL Code Used

- none

Call Back Group 2

Script File Name BP Call Back Group 2.script

Brief Description

Call Back Group 2 is the second of the three sample groups provided to demonstrates a mechanism for capturing information to schedule a call back. It is similar to the first call back sample allowing the user to select whether the call back should be scheduled for a specific day of the week or for a specific date. It is different in that it captures the preferred date and time of day to schedule the call back using text entry instead of drop down controls. It also allows for capturing any notes that will assist in handling the call back.

High Level Flow

If the user selects on the first panel of this group to schedule the call back for a specific day of the week, the next panel that will be displayed will have only one answer node which is a radio button control for selecting the desired day of the week (Monday, Tuesday, Wednesday, etc.).

If the user selects on the first panel of this group to schedule the call back for a specific date, the next panel that will be displayed will have only one answer node which is defined as a text field and is used to enter the month, day and year to schedule the call back. The valid date format is MM/DD/YYYY. The text field will display a default value of the current day's date which can be overridden by the user. The script will validate the date format and that the entered date is not a date in the past. Modifications to the default and validation logic will require changes to the supporting Java code.

After the specific day of week or date information has been captured, the time of day panel will be displayed. This panel has only one answer node which is defined as a text field and is used to enter the preferred time of day for the call back. The valid time format is HH:MM:SS[AM|PM]. The text field will display a default value of the current time which can be overridden by the user. The script will validate that a valid time format has been entered. Modifications to the default and validation logic will require changes to the supporting Java code.

Finally, the script flows to the last panel in the group to capture call back notes in a text area control.

Java Source Code Used

- BPUtilities.java

Database Tables/Views Used

- none

PL/SQL Code Used

- none

Call Back Group 3

Script File Name BP Call Back Group 3.script

Brief Description

Call Back Group 3 is the final of the three sample groups provided to demonstrates a mechanism for capturing information to schedule a call back. It is different from the first and second call back samples in that it does not provide the user with the option to select whether the call back should be scheduled for a specific day of the week or for a specific date. Instead the script consists of three panels that flow by default from one panel to the next. The first panel captures the best time to call. The second panel captures the number to call back. And the third panel captures any notes that will assist in handling the call back.

High Level Flow

This group consists of three panels. The first panel has just a single answer node which is defined as a drop down control and comes populated with six typical lookup values specified for the best time to call time ranges:

- Weekday 8AM - 12PM
- Weekday 12PM - 5PM
- Weekday 5PM - 8PM
- Weekend 8AM - 12PM
- Weekend 12PM - 5PM
- Weekend 5PM - 8PM

The list of lookup values can be easily modified to meet the needs of your implementation by editing the data dictionary for the answer node.

The second panel of this group has three answer nodes each defined as a text field. The first two answer nodes capture the area code and phone number of the number to be used in the call back. Both of these are required fields. The script has built in logic to verify that data has been entered into these fields. The third answer node captures the extension, if any, of the phone number to be used in the call back. This field is not required.

The third and final panel in this group has a single answer node which is used to capture call back notes in a text area control.

Java Source Code Used

- BPUtilities.java

Database Tables/Views Used

- none

PL/SQL Code Used

- none

Do Not Call Group

Script File Name BP Do Not Call Group.script

Brief Description

The Do Not Call Group is an example of one way to handle a customer or prospect who has requested not to be called again.

High Level Flow

This group consists of four panels. The first panel provides some sample text of what is appropriate language for a Do Not Call request. The panel has two button controls. One button allows the user to cancel the request in case they have navigated to this group by accident or if the customer has changed their mind. The other button allows the user to continue with the request.

On the second panel, the customer is asked if they would like the address of the Direct Marketing Association Telephone Preference Service. If the customer accepts the offer then the next panel will display the DMA address and then will continue on to the closing panel. If the customer does not wish to get the DMA address then the script will flow directly to the closing panel. The closing panel embeds the appropriate time of day into the panel text using a Java command.

Java Source Code Used

- BPUtilities.java

Database Tables/Views Used

- none

PL/SQL Code Used

- none

Validation Utilities

Script File Name BP Validation Utilities.script

Brief Description

This script demonstrates several common validation routines that are used throughout the Best Practices scripts. It is likely that you will want to use some validations of this sort in the scripts you develop for your implementation. Sample data validations are included for:

- required field
- date format
- time format
- integer data type
- integer within a specified range

Also demonstrated are:

- the use of a default value set to the current date and time
- panel text Embedded Value display of correct time of day for greeting

High Level Flow

This script consists of a single panel with five answer nodes. Each answer node demonstrates a different type of text field validation. Each validation displays an appropriately formatted message if the field validation fails. The user is allowed to correct the text field entry and retry the validation.

The first answer node demonstrates the validation of a text field that must be in a date format of MM/DD/YYYY. In addition, the date must not be in the past. The text field is defaulted to the current day's date.

The second answer node demonstrates the validation of a text field that must be in a time format of HH:MM:SS [AM|PM]. The text field is defaulted to the current time.

The third answer node demonstrates the validation of a required text field.

The fourth answer node demonstrates the validation of a required text field that must be of an integer value.

The fifth and final answer node demonstrates the validation of a required text field that must be of an integer value and be within a range of a minimum value and maximum value.

Java Source Code Used

- BPUtilities.java

Database Tables/Views Used

- none

PL/SQL Code Used

- none

Customer Data Source Access Scripts

Customer Data Source Overview

The following series of scripts provide examples of how data from the Oracle Customer model can be accessed for read only querying purposes or for updating purposes. You can use these scripts as a starting point to other scripts you may want to develop that need to access customer data.

All of these script examples assume that the script will be integrated to a front end application that will provide some key data values such as the Customer Id or the Customer Account Id.

All of these scripts depend on Java code to function properly. The source Java code has been included with this release of the Scripting Best Practices so you can make modifications to it to meet your own implementation requirements. It is recommended that you keep the original source code intact, making a copy of it to use as your starting point for any variation you want to develop. In addition to the source code, the iessampl.jar file is included with this release. This jar file is the archive file for the seven java source files that are used by the Customer Data Source Access scripts.

Four of the six scripts in this set perform updates to the Customer data model. These scripts use PL/SQL commands to update the data. The PL/SQL APIs that support these functions have also been included in the release.

Customer/Prospect Update

Script File Name BPPROS.script

Brief Description

This script queries a customer or prospect record and displays some basic information allowing the user to update this information.

The script expects to have an the following information passed to it to use as its query selection criteria:

- Customer Id

The following customer/prospect information can be updated:

- Title
- First Name
- Middle Name
- Last Name
- Suffix
- Nickname
- Social Security Number
- Date of Birth
- Gender
- Phone Line Type
- Country Code
- Area Code
- Phone Number
- Extension
- Employed As Title
- Employed By Division Name
- Employed By Company Name

High Level Flow

1. The database is queried using the Customer Id retrieving the customer record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

2. The customer data is displayed in a series of answer nodes defined as text fields. The user may modify the data here to update the customer record in the database.

3. The script, using the supporting Java code, determines if the Update transaction was successful.

   If the Update transaction failed, an appropriate message is displayed and the script skips to the Call Wrap Up.

4. Transaction confirmation message is displayed in the panel text.

5. The script continues performing a query to retrieve all the phone number records for this customer.

   If no matching records are found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

6. If one or more matching records are found, then the phone numbers are displayed in an answer node defined as a drop down list. The user must select one number from the list.

7. The database is queried using the phone number selected retrieving the customer phone number record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

8. The customer phone data is displayed in a series of answer nodes defined as text fields. The user may modify the data here to update the customer phone record in the database.

9. The script, using the supporting Java code, determines if the Update transaction was successful.

   If the Update transaction failed, an appropriate message is displayed and the script skips to the Call Wrap Up.

10. Transaction confirmation message is displayed in the panel text and the user is asked if they would like to update another phone record.

11. If the user selects to update another phone record, the script will loop back to Step 5 and perform another query to retrieve all the phone records for this customer. The script will continue to flow through as stated above.
12. If the user does not select to update another phone record, the script will continue performing a query to retrieve all the employment history records for this customer.

    If no matching records are found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.
13. If one or more matching records are found, then the employment titles are displayed in an answer node defined as a drop down list. The user must select one title from the list.
14. The database is queried using the employment title selected retrieving the customer employment history record values if a matching record is found.

    If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.
15. The customer employment history data is displayed in a series of answer nodes defined as text fields. The user may modify the data here to update the customer employment information in the database.
16. The script, using the supporting Java code, determines if the Update transaction was successful.

    If the Update transaction failed, an appropriate message is displayed and the script skips to the Call Wrap Up.
17. Transaction confirmation message is displayed in the panel text.
18. Call Wrap Up (this is just a place holder; the Call WrapUp group of your choice should be substituted here)
19. The script is terminated.

Java Source Code Used

- BPBase.java
- BPProspects.java

Database Tables/Views Used

- HZ_PARTIES
- HZ_PERSON

- JTF_CONTACT_POINTS_V
- HZ_EMPLOYMENT_HISTORY

PL/SQL Code Used

- iessmpls.pls
- iessmplb.pls
- IES_SAMPLES_PKG.UPDATE_PERSON
- IES_SAMPLES_PKG.UPDATE_CONTACT_POINT
- IES_SAMPLES_PKG.UPDATE_EMPLOYMENT

Customer Address Update

Script File Name BPADDR.script

Brief Description

This script queries a customer primary address record and displays some basic information about the address allowing the user to update this information.

The script expects to have an the following information passed to it to use as its query selection criteria:

- Customer Id
- Customer Site Id
- Location Id

The following information can be updated:

- Address Type
- Address Line 1
- Address Line 2
- Address Line 3
- Address Line 4
- City
- State
- Province
- Postal Code
- Country

High Level Flow

1. The database is queried using the Customer Id, Customer Site Id and Location Id retrieving the customer's primary address record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

2. The primary address data is displayed in a series of answer nodes defined as text fields. The user may modify the data here to update the address record in the database.
3. The script, using the supporting Java code, determines if the Update transaction was successful.

   If the Update transaction failed, an appropriate message is displayed and the script skips to the Call Wrap Up.
4. Transaction confirmation message is displayed in the panel text.
5. Call Wrap Up (this is just a place holder; the Call WrapUp group of your choice should be substituted here)
6. The script is terminated.

Java Source Code Used

- BPBase.java
- BPAddress.java

Database Tables/Views Used

- JTF_PARTY_ADDRESSES_V

PL/SQL Code Used

- iessmpls.pls
- iessmplb.pls
- IES_SAMPLES_PKG.UPDATE_ADDRESS

Customer Billing Address Query

Script File Name BPBILLADDR.script

Brief Description

This script queries a customer’s billing address record and displays some basic information about the address. The script does not allow the user to update this information.

The script expects to have an the following information passed to it to use as its query selection criteria:

- Customer Id
- Customer Site Id
- Location Id

The following information is displayed:

- Address Type
- Address Line 1
- Address Line 2
- Address Line 3
- Address Line 4
- City
- State
- Province
- Postal Code
- Country

High Level Flow

1. The database is queried using the Customer Id, Customer Site Id and Location Id retrieving the customer’s billing address record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

2. The billing address data is displayed as embedded values in the panel text
3. Query confirmation message is displayed in the panel text.
4. Call Wrap Up (this is just a place holder; the Call WrapUp group of your choice should be substituted here)
5. The script is terminated.

Java Source Code Used

- BPBase.java
- BPBillAddress.java

Database Tables/Views Used

- JTF_PARTY_ADDRESSES_V

PL/SQL Code Used

- None

Customer Fulfillment Address Update

Script File Name BPFMADDR.script

Brief Description

This script queries a customer collateral address record and displays some basic information about the address allowing the user to update this information.

The script expects to have an the following information passed to it to use as its query selection criteria:

- Customer Id
- Customer Site Id
- Location Id

The following information can be updated:

- Address Type
- Address Line 1
- Address Line 2
- Address Line 3
- Address Line 4
- City
- State
- Province
- Postal Code
- Country

High Level Flow

1. The database is queried using the Customer Id, Customer Site Id and Location Id retrieving the customer's collateral address record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

2. The collateral address data is displayed in a series of answer nodes defined as text fields. The user may modify the data here to update the address record in the database.
3. The script, using the supporting Java code, determines if the Update transaction was successful.

   If the Update transaction failed, an appropriate message is displayed and the script skips to the Call Wrap Up.
4. Transaction confirmation message is displayed in the panel text.
5. Call Wrap Up (this is just a place holder; the Call WrapUp group of your choice should be substituted here)
6. The script is terminated.

Java Source Code Used

- BPBase.java
- BPFMAddress.java

Database Tables/Views Used

- JTF_PARTY_ADDRESSES_V

PL/SQL Code Used

- iessmpls.pls
- iessmplb.pls
- IES_SAMPLES_PKG.UPDATE_ADDRESS

Customer Account Query

Script File Name BPACCNT.script

Brief Description

This script queries a customer account record and displays some basic information about the account. It does not allow for account updates.

The script expects to have an Account Id passed to it to use as its query selection criteria.

The following information is displayed:

- Customer Account Id
- Customer Id
- Customer Type
- Account Name
- Original System Reference
- Password Text
- PIN
- Account Status

High Level Flow

1. The database is queried using the Account Id, retrieving the account record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

2. The account data is displayed as embedded values in the panel text.
3. Query confirmation message is displayed in the panel text.
4. Call Wrap Up (this is just a place holder; the Call WrapUp group of your choice should be substituted here)
5. The script is terminated.

Java Source Code Used

- BPBase.java

- BPAccount.java

Database Tables/Views Used

- JTF_CUST_ACCOUNTS_V

PL/SQL Code Used

- None

Customer Account Suspension Update

Script File Name BPSUSPEND.script

Brief Description

This script queries a customer account record and displays some basic information about the account. It does allow for the account suspension date to be updated.

The script expects to have an Account Id passed to it to use as its query selection criteria.

The following information is displayed:

- Customer Account Id
- Customer Id
- Customer Type
- Account Name
- Original System Reference

The following information can be updated:

- Suspension Date

High Level Flow

1. The database is queried using the Account Id, retrieving the account record values if a matching record is found.

   If no matching record is found (i.e. the query is invalid), an appropriate message is displayed and the script skips to the Call Wrap Up.

2. Most of the account data is displayed as embedded values in the panel text. The Account Suspension Date, if any, is displayed in an answer node defined as a text field. The user may enter a date here to update the account record in the database.
3. The script, using the supporting Java code, determines if the Update transaction was successful.

   If the Update transaction failed, an appropriate message is displayed and the script skips to the Call Wrap Up.

4. Transaction confirmation message is displayed in the panel text.

5. Call Wrap Up (this is just a place holder; the Call WrapUp group of your choice should be substituted here)
6. The script is terminated.

Java Source Code Used

- BPBase.java
- BPSuspend.java

Database Tables/Views Used

- JTF_CUST_ACCOUNTS_V

PL/SQL Code Used

- iessmpls.pls
- iessmplb.pls
- IES_SAMPLES_PKG.UPDATE_CUST_ACCT_SUSPENSION

Scripting Best Practices Java Code

This document includes information on the following Java files:

- BPUtilities.java
- BPBase.java
- BPProspect.java
- BPAccount.java
- BPAddress.java
- BPBillAddress.java
- BPFMAddress.java
- BPSuspend.java

Java Files Overview

Many of the Best Practice scripts use supporting Java commands. If you want to change or add to the Java commands you will need to have some Java programming experience. However if you know that a specific piece of logic already exist as a Java command in one script you can easily reuse it within the same script or in another script without changing the underlying Java code. All you need to do is specify the correct Java command and parameter information in the script.

This document provides descriptions for each of the Java commands that are used by the Best Practices scripts. The required parameter information is given with each command description. If the command uses the Oracle Scripting ServerProxyBean, then the command should be specified with a parameter name of ‘Proxy’.

The Java source code files have been included in the release so they can be copied and modified to meet your needs. The compiled class and jar files have also been included in the release for ease of use if your implementation does not require any modifications.

BPUtilities.java

Brief Description

This Java class is used to perform several common validation routines that are used throughout the Best Practices scripts. It is likely that you will want to use some validations of this sort in the scripts you develop for your implementation.

Data validations are included for:

- required field
- date format
- date not in the past
- time format
- integer data type
- integer within a specified range

This code is also used to determine:

- the current date and time
- the time of day for greeting and closing text
- This code also defines a Shortcut button for:
- Customer Search
- Name Verification
- Address Verification

Related Java Files

- BPUtilities.class

Used by

- BP Validation Utilities.script
- BP Call Back Group 2.scipt
- BP Call Back Group 3.script
- BP Do Not Call Group.script

Java Commands Included

validateRequiredField Enforces a value is entered in an Answer defined as a text field.

If the validation fails, the following error message is displayed:

“Please enter a value for: ‘label’ ”

Parameters:

String answer

String label

validateForDateFormat Enforces a date string in the format: MM / DD / YYYY.

If the validation fails, the following error message will be displayed:

“Please enter a valid date in the specified format: MM / DD / YYYY”.

Parameters:

String answer

validateDateNotInPast Enforces a date string that does not specify a date in the past.

If the validation fails, the following error message will be displayed:

“Please enter a valid date / time that is not in the past.”

Parameters:

String answer

validateForTimeFormat Enforces a time string in the format: HH:MM:SS [AM | PM].

If the validation fails, the following error message will be displayed:

“Please enter a valid time in the specified format: HH:MM:SS [AM | PM]”.

Parameters:

String answer

validateIntegerEntered Enforces an integer value is entered in an Answer defined as a text field.

If the validation fails, the following error message is displayed:

“Please enter an integer for: ‘label’ ”

Parameters:

String answer

String label

validateIntegerInRange Enforces an integer value within a specified range (minStr - maxStr) is entered in an Answer defined as a text field.

If the validation fails, the following error message is displayed:

“Please enter an integer between ‘min’ and ‘max’ for: ‘label’ ”

Parameters:

String answer

String minStr

String maxStr

String label

getTimeOfDayForGreeting Looks at the local time and returns the correct time of day for the greeting panel.

If (hour < 12), returns “morning”;

if (hour < 18), returns “afternoon”;

else, returns “evening”.

No parameters.

getTimeOfDayForClosing Looks at the local time and returns the correct time of day for the closing panel.

If (hour < 12), returns “day”;

if (hour < 17), returns “afternoon”;

else, returns “night”.

No parameters.

shortcutCustomerSearch Jumps to a Group with a Shortcut name of ‘CustomerSearch’.

Uses the Oracle Scripting ServerProxyBean.

shortcutNameVerification Jumps to a Group with a Shortcut name of ‘NameVerification’.

Uses the Oracle Scripting ServerProxyBean.

shortcutAddressVerification Jumps to a Group with a Shortcut name of ‘AddressVerification’.

Uses the Oracle Scripting ServerProxyBean.

BPBase.java

Brief Description

This Java class is used by all of the Best Practices scripts that perform data access routines. It handles the common script initialization functions that allow Oracle Scripting to startup with all the information required to directly integrate with the Oracle CRM and Oracle Applications infrastructure components.

Related Java Files

- BPBase.class
- iessampl.jar

Used by

- BPPROS.script
- BPADDR.script
- BPBILLADDR.script
- BPFMADDR.script
- BPACCNT.script
- BPSUSPEND.script

Java Commands Included

getNextCall Script preaction command used to integrate with CTI technology. This command uses the appsInitialize command (see below) to get the primary connection information. It sets the initial blackboard values for the Customer Id, transaction variables, and message variables.

Uses the Oracle Scripting ServerProxyBean.

appsInitialize Calls a PL / SQL package which sets the client application id, responsibility id, user id and security group id in the AOL object library for this JDBC connection / database session.

Parameters:

Integer UserId

Integer RespId

Integer AppId

Integer SecGroupID

Connection Connection

wrapUpCall Script wrap up action for CTI technology. Currently this command is just a stub place holder. Future Best Practice release will include sample processing here.

Uses the Oracle Scripting ServerProxyBean.

validateRequiredAnswer Enforces a value is entered in an Answer defined as a text field.

If the validation fails, the following error message is displayed:

"Please enter all required fields."

Parameters:

String answer

cursorIsValid Returns an indicator that specifies if the SQL cursor is valid. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Returns an indicator that specifies if the transaction is valid. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

safeSetBlackBoardAnswer Sets a null value retrieved from the database to a space to prevent a NullPointerException error from occurring when attempting to copy the value to the BlackBoard.

Uses the Oracle Scripting ServerProxyBean.

Parameters:

String key

BPProspects.java

Brief Description

This Java class is used by the Best Practices script that accesses the basic customer demographic data. It initializes the BlackBoard key values for the transaction status variables and all the data elements that are retrieved from the database during the script.

Related Java Files

- BPProspects.class
- iessampl.jar

Used by

- BPPROS.script

Java Commands Included

moveTransactValuesToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from the HZ_PARTIES and HZ_PERSON_PROFILES tables during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

moveContactPointToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from the JTF_CONTACT_POINTS_V view during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

moveEmploymentHistoryToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from the HZ_ EMPLOYMENT_HISTORY table during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

getNextCall Uses BPBase getNextCall command.

Uses the Oracle Scripting ServerProxyBean.

wrapUpCall Uses BPBase wrapUpCall command.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Uses BPBase transactionIsValid command. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

continueCheck Returns an indicator that specifies if a continuation button is Y. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

BPAccount.java

Brief Description

This Java class is used by the Best Practices script that accesses the customer account data. It initializes the BlackBoard key values for the transaction status variables and all the data elements that are retrieved from the database during the script.

Related Java Files

- BPAccount.class
- iessampl.jar

Used by

- BPACCNT.script

Java Commands Included

moveTransactValuesToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from the JTF_CUST_ACCOUNTS_V view during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

getNextCall Uses BPBase getNextCall command. It sets the initial blackboard value for the Account Id.

Uses the Oracle Scripting ServerProxyBean.

wrapUpCall Uses BPBase wrapUpCall command.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Uses BPBase transactionIsValid command. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

BPAddress.java

Brief Description

This Java class is used by the Best Practices script that accesses the customer address data. It initializes the BlackBoard key values for the transaction status variables and all the data elements that are retrieved from the database during the script.

Related Java Files

- BPAddress.class
- iessampl.jar

Used by

- BPADDR.script

Java Commands Included

moveTransactValuesToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from JTF_PARTY_ADDRESSES_V view during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

getNextCall Uses BPBase getNextCall command. It sets the initial blackboard values for the Customer Site Id and Location Id.

Uses the Oracle Scripting ServerProxyBean.

wrapUpCall Uses BPBase wrapUpCall command.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Uses BPBase transactionIsValid command. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

BPBillAddress.java

Brief Description

This Java class is used by the Best Practices script that accesses the customer billing address data. It initializes the BlackBoard key values for the transaction status variables and all the data elements that are retrieved from the database during the script.

Related Java Files

- BPBillAddress.class
- iessampl.jar

Used by

- BPBILLADDR.script

Java Commands Included

moveTransactValuesToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from JTF_PARTY_ADDRESSES_V view during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

getNextCall Uses BPBase getNextCall command. It sets the initial blackboard values for the Customer Site Id and Location Id.

Uses the Oracle Scripting ServerProxyBean.

wrapUpCall Uses BPBase wrapUpCall command.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Uses BPBase transactionIsValid command. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

BPFMAddress.java

Brief Description

This Java class is used by the Best Practices script that accesses the customer collateral mailing address data. It initializes the BlackBoard key values for the transaction status variables and all the data elements that are retrieved from the database during the script.

Related Java Files

- BPFMAddress.class
- iessampl.jar

Used by

- BPFMADDR.script

Java Commands Included

moveTransactValuesToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from JTF_PARTY_ADDRESSES_V view during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

getNextCall Uses BPBase getNextCall command. It sets the initial blackboard values for the Customer Site Id and Location Id.

Uses the Oracle Scripting ServerProxyBean.

wrapUpCall Uses BPBase wrapUpCall command.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Uses BPBase transactionIsValid command. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

BPSuspend.java

Brief Description

This Java class is used by the Best Practices script that accesses the customer account data for editing the Suspension Date. It initializes the BlackBoard key values for the transaction status variables and all the data elements that are retrieved from the database during the script.

Related Java Files

- BPSuspend.class
- iessampl.jar

Used by

- BPSUSPEND.script

Java Commands Included

moveTransactValuesToBlackboard Uses the safeSetBlackBoardAnswer command from BPBase class to set any null values retrieved from the database to a space. Repeats the command for each data element retrieved from the JTF_CUST_ACCOUNTS_V view during the script. Used as a Preaction on a Panel that displays database values retrieved.

Uses the Oracle Scripting ServerProxyBean.

getNextCall Uses BPBase getNextCall command. It sets the initial blackboard value for the Account Id.

Uses the Oracle Scripting ServerProxyBean.

wrapUpCall Uses BPBase wrapUpCall command.

Uses the Oracle Scripting ServerProxyBean.

transactionIsValid Uses BPBase transactionIsValid command. Used for Conditional Branching.

Uses the Oracle Scripting ServerProxyBean.

Scripting Best Practices PL/SQL Code

This document includes information on the following PL/SQL files:

- iessmplb.pls
- iessmpls.pls

PL/SQL Files Overview

Several of the Best Practice scripts use supporting PL/SQL commands to execute the PL/SQL APIs that update the database. If you want to change or add to the PL/SQL commands you will need to have some PL/SQL programming experience and a good understanding of the data model. However if you know that certain pieces of data are already being manipulated by some existing PL/SQL commands in one script, you can reuse it within the same script or in another script without changing the underlying PL/SQL code. The easiest way to reuse the PL/SQL commands is to identify the script objects that contain them and then either copy those objects or import them from another script. This document provides descriptions for each of the PL/SQL commands that are used by the Best Practices scripts.

The PL/SQL source code files have been included in the release so they can be copied and modified to meet your needs. The two PL/SQL source code files that are used by the Best Practices scripts are IESSMPLB.PLS and IESSMPLS.PLS. We use two files because of the unique architecture of the Oracle CRM/Applications public API standards. These standards imply the usage of PL/SQL record types, which are not supported by the Oracle JDBC drivers, so a PL/SQL interface layer between Oracle Scripting and the Oracle CRM/Applications public APIs has been created. The interface layer basically “flattens-out” the record types into individual API parameters.

This PL/SQL interface layer is called using a PL/SQL-based, Oracle Scripting command. Using the PL/SQL-based, Oracle Scripting command, the PL/SQL interface layer API parameters are defined as parameters within the command definition dialog and the blackboard keys are specified as the parameter values. This approach has a lot of merit because it is more of a pure Oracle Scripting solution. This is the approach demonstrated in the Best Practices scripts.

Another alternative to this approach would be to call this PL/SQL interface layer using the Oracle JDBC technology executed from a Java-based Oracle Scripting command. In the Java-based Oracle Scripting command, the script author would be required to develop custom Java code to use the PL/SQL interface layer, but offers a little more control over the transaction processing and exception handling. The code might be easier to reuse and maintain as the code is not copied from one script to another. This approach is not demonstrated in the Best Practices scripts.

iessmpls.pls

Brief Description

This file contains the package specification for the IES_SAMPLES_PKG package. These are the API functions used to update some customer, phone, address, employment and account information as defined in the Oracle Scripting Best Practices scripts. These APIs provide an interface to the Oracle Scripting Best Practice scripts and the public APIs available in the Oracle Receivables TCA Customer Model.

Related files

iessmplb.pls

Used by

- BPPROS.script
- BPADDR.script
- BPBILLADDR.script
- BPFMADDR.script
- BPACCNT.script
- BPSUSPEND.script

PL/SQL Procedures Included

update_person Updates the demographic information for the party identifier supplied.

```
PROCEDURE update_person (
       p_party_id               IN  NUMBER
      ,p_profile_id             IN  NUMBER
      ,p_title                  IN  VARCHAR2
      ,p_first_name             IN  VARCHAR2
      ,p_middle_name            IN  VARCHAR2
      ,p_last_name              IN  VARCHAR2
      ,p_suffix                 IN  VARCHAR2
      ,p_nick_name              IN  VARCHAR2
      ,p_ss_number              IN  VARCHAR2
      ,p_date_of_birth          IN  VARCHAR2
      ,p_gender                 IN  VARCHAR2
      ,p_commit                 IN  VARCHAR2
      ,x_return_status          OUT VARCHAR2
      ,x_msg_count              OUT NUMBER
      ,x_msg_data               OUT VARCHAR2
      );
```

update_contact_point Updates contact point information for the party and contact point identifiers supplied.

```
PROCEDURE update_contact_point (
        p_party_id              IN  NUMBER
       ,p_contact_point_id      IN  NUMBER
       ,p_phone_line_type       IN  VARCHAR2
       ,p_country_code          IN  VARCHAR2
       ,p_area_code             IN  VARCHAR2
       ,p_phone_number          IN  VARCHAR2
       ,p_extension             IN  VARCHAR2
       ,p_commit                IN  VARCHAR2
       ,x_return_status         OUT VARCHAR2
       ,x_msg_count             OUT NUMBER
       ,x_msg_data              OUT VARCHAR2
       );
```

update_employment Updates the employment history information for the party and employment history identifiers supplied.

```
PROCEDURE update_employment (
        p_party_id                  IN  NUMBER
       ,p_employment_history_id     IN  NUMBER
       ,p_employed_as_title         IN  VARCHAR2
       ,p_employed_by_division_name IN  VARCHAR2
       ,p_employed_by_name_company  IN  VARCHAR2
       ,p_commit                    IN  VARCHAR2
       ,x_return_status             OUT VARCHAR2
       ,x_msg_count                 OUT NUMBER
       ,x_msg_data                  OUT VARCHAR2
       );
```

update_address Updates the address information for the party, party site, and location identifiers supplied.

```
PROCEDURE update_address (
        p_location_id           IN  NUMBER
       ,p_address1              IN  VARCHAR2
       ,p_address2              IN  VARCHAR2
       ,p_address3              IN  VARCHAR2
       ,p_address4              IN  VARCHAR2
       ,p_city                  IN  VARCHAR2
       ,p_state                 IN  VARCHAR2
       ,p_province              IN  VARCHAR2
       ,p_postal_code           IN  VARCHAR2
       ,p_country               IN  VARCHAR2
       ,p_commit                IN  VARCHAR2
       ,x_return_status         OUT VARCHAR2
       ,x_msg_count             OUT NUMBER
       ,x_msg_data              OUT VARCHAR2
       );
```

update_cust_acct_suspension Updates the service suspension date for the account identifier supplied.

```
PROCEDURE update_cust_acct_suspension (
        p_party_id              IN  NUMBER
       ,p_cust_account_id       IN  NUMBER
,p_customer_type         IN  VARCHAR2
,p_account_name          IN  VARCHAR2
,p_org_system_reference  IN  VARCHAR2
,p_suspension_date       IN  VARCHAR2
,p_commit                IN  VARCHAR2
,x_return_status         OUT VARCHAR2
,x_msg_count             OUT NUMBER
,x_msg_data              OUT VARCHAR2
);
```

iessmplb.pls

Brief Description

This file contains the package body for the IES_SAMPLES_PKG package. These are the API functions used to update some customer, phone, address, employment and account information as defined in the Oracle Scripting Best Practices scripts. These APIs use the public APIs available in the Oracle Receivables TCA Customer Model.

Related files

iessmpls.pls

Used by

- BPPROS.script
- BPADDR.script
- BPBILLADDR.script
- BPFMADDR.script
- BPACCNT.script
- BPSUSPEND.script

PL/SQL Procedures Included

The same five procedures as defined in the iessampls.pls description above are included in this file. However, the detail that is involved is much greater and beyond the scope of the Best Practices scripts. Please refer to the source code listing for this file and to the API documentation that is included with the Oracle Receivables TCA Customer Model.