Oracle® Scripting

User Guide

Release 12

Part No. B28102-02

October 2006

ORACLE®

Oracle Scripting User Guide, Release 12

Part No. B28102-02

Primary Author: Michael Meditzky

Contents

3 Understanding the Scripting Engine

4 Understanding the Scripting Administration Console

5 Understanding Survey Campaigns

6 Planning Oracle Scripting Projects

7 Using the Script Wizard

8 Using Graphical Scripts

9 Using the Scripting Engine

A Example of Starting to Create a Wizard Script

Glossary

Send Us Your Comments

Oracle Scripting User Guide, Release 12

Part No. B28102-02

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on Oracle MetaLink and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

Intended Audience

Welcome to Release 12 of the *Oracle Scripting User Guide.*

This guide is intended for users of one or more of the components of Oracle Scripting.

This guide assumes that you have a working knowledge of the following:

- The principles and customary practices of your business area
- Oracle Scripting
- The Oracle Applications graphical user interface

  To learn more about the Oracle Applications graphical user interface, read the *Oracle Applications User's Guide*.

See Related Information Sources on page xiv for more Oracle Applications product information.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to

evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/ .

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

Related Information Sources

Oracle Scripting Implementation Guide

This guide describes how to implement Oracle Scripting components, how to administer Oracle Scripting, and how to test an implementation appropriately.

Oracle Scripting Developer's Guide

This guide is intended for advanced users of Script Author. It includes information on the various Script Author commands, a description of Best Practice Java methods that you can use to extend your scripts, and how to perform advanced tasks in a graphical script. This document also lists the seeded reusable commands, describes each Oracle Scripting building block, and explains how to customize panel layouts.

Oracle Application Framework Personalization Guide

This guide describes how to personalize Oracle Application Framework-based Oracle E-Business Suite application pages as an end-user and as a personalization administrator using the Oracle Application Personalization Framework.

Note: Oracle Scripting does not contain any end-user personalizable regions, and there are no special considerations that you need to be aware of when creating administrator-level personalizations of its regions or pages. For general information about how to create personalizations, refer to the *Oracle Application Framework Personalization Guide*.

Oracle Application Framework Documentation

For more information about Oracle Application Framework documentation, refer to *Oracle Application Framework Documentation Resources*, MetaLink Note 391554.1.

Integration Repository

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who

has changed your information because SQL*Plus and other database tools do not keep a record of changes.

1

Introduction to Oracle Scripting

This chapter covers the following topics:

- Overview of Oracle Scripting
- Components
- How Scripts are Used

Overview of Oracle Scripting

Oracle Scripting is a set of tools that provides enterprises with scripts, which guide customers, agents, and employees through decision flows based on a series of questions and answers. Through branching logic built into the product, script flow is determined both by customer answers and by data stored in the customer tables.

Scripts can be used for a variety of purposes:

- To assist enterprise agents in gathering data from or providing information to customers and prospects
- As Web-based surveys to solicit information from a sample or target population
- As self-service Web scripts that integrate with enterprise Web pages and provide a mechanism to guide Web customers through decision processes

Components

Oracle Scripting is composed of four components: Script Author, the Scripting Engine, the Scripting Administration console, and the Survey Administration console.

1. Using Script Author, trained script developers translate business requirements into miniature programs called scripts. Script Author supports two methods to create a script: using graphical layout tools to create graphical scripts, and using a Script Wizard feature to create wizard scripts. Both script types can be executed in either

of the two Scripting Engine interfaces.

2. Using the Scripting Engine component, script end users execute the script in one of two runtime interfaces:

   - The Scripting Engine agent interface is used in combination with Oracle business applications. Users of the agent interface are interaction center agents; they launch each script - which executes as a Java applet - from an Oracle business application form.

   - Using the Scripting Engine Web interface, scripts execute in any Oracle Applications 12 certified Web browser. These can be used as survey questionnaires, or as Web scripts launched from an integrated Oracle self-service Web application such as Oracle iSupport.

     Note: Any script executed in the Scripting Engine Web interface, either as a survey questionnaire or as a Web script, requires a survey campaign to be defined and one of its deployments activated.

3. Using the Scripting Administration console, script administrators can launch Script Author as a Java applet, administer Oracle Scripting script files and custom Java archive files, and view panel footprint summary reports.

4. Using the Survey Administration console, you administer survey campaigns and associated survey resources.

   Survey resources can include header and footer sections, error pages and final pages. For each script executed in the Web interface, the appropriate survey resources display.

The following summarizes the main usages of each component:

- Use Script Author to build a script for execution in either Scripting Engine interface.

- Use the Scripting Administration console to access Script Author (as a Java applet), to manage scripting files or custom Java archives, or to view panel footprint summary reports.

- Use the Survey Administration console to administer survey campaigns, which is prerequisite to executing any script in a Web browser.

- Use Scripting Engine to execute any script. The two interfaces include the agent interface (a Forms client) or the Web interface (a Web browser).

How Scripts are Used

There are various ways in which scripts can be employed to gather or distribute data for an enterprise. For example:

- Scripts in the Scripting Engine agent interface can be launched from Oracle TeleSales and the Customer Support component of Oracle TeleService.

  Note: Use of the Scripting Engine agent interface in standalone mode (without other business applications) is not supported by Oracle except for script testing and validation.

- A script can serve as a survey questionnaire to solicit specific information from the sample or target population. Such a questionnaire can be executed in the Scripting Engine agent interface, or set up as a survey campaign and executed in a Web browser in the Scripting Engine Web interface.

- Scripts or surveys can be executed in the Web interface as follows:

  - An active survey deployment URL can be embedded as a hyperlink in an enterprise Web site pointing to a survey deployment.

  - An active survey deployment URL can be embedded as a hyperlink in an Oracle self-service Web application (e.g., Oracle iSupport or Oracle iStore). Users of the Web application can launch a script in the browser. When executed in this manner, the script is referred to as a Web script. Web scripts are a valuable resource to provide scripted information to Web application users or to solicit feedback to the enterprise regarding the application user's experience.

  - By leveraging other Oracle Applications (Oracle One-to-One Fulfillment and Oracle Marketing Online), you can send e-mail invitations and reminders to participate in a survey questionnaire to members of a defined population or sample, as set up on an Oracle Marketing list. The list includes a link to an active survey deployment.

2

Understanding Script Author

This chapter covers the following topics:

- Introduction
- Script Author Terminology
- Script Author Concepts and Features
- Scripts
- Graphical Scripts
- Graphical Script Components
- Global Script Properties
- Hierarchical Graphs
- Wizard Scripts
- Comparisons Between Graphical and Wizard Scripts
- Minimum Requirements for Any Graph
- Oracle Scripting Users
- Planning Scripts
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Script Author File Management

Introduction

Script Author is the component of Oracle Scripting used to create, modify, and deploy scripts. These scripts can be executed using the Scripting Engine component of Oracle Scripting. Script Author is a Java applet intended for use by trained functional users.

You can create both graphical scripts and wizard scripts using the Script Author. Wizard scripts do not contain all the features available for graphical scripts, but provide a useful startup function for novice users.

You can also start a script application by creating a wizard script, and create a graphical script from the wizard script. You can then add in the extra features and custom code that you require into the graphical script.

To access the Script Author Java applet, you must log into Oracle Applications with a user account that is assigned the Scripting Administrator responsibility. This results in the appearance of the Scripting Administration console. From the Home tab, click Launch Script Author. A separate Oracle JInitiator window appears. Subsequently, Script Author executes as a Java applet from the Oracle JInitiator session, using the current logged-in user's database authentication information.

Script Author Terminology

Following are definitions of some terms helpful in understanding Script Author. For more terminology, refer to the glossary, page Glossary-1.

script

A script is a miniature program built using Script Author to facilitate the flow of information between an enterprise and other parties. Each script enforces business rules programmed into it by script developers. Script Author provides the ability to create two kinds of scripts: wizard scripts and graphical scripts. Scripts can be executed in any Scripting Engine runtime interface.

graphical script

A graphical script is any script created or modified using the graphical tools of Script Author. Graphical scripts are created, edited and deployed using the Script Author Java applet.

wizard script

A wizard script is a script created using the Script Wizard feature of Script Author. Like graphical scripts, wizard scripts are executed in any Scripting Engine interface. By answering questions in a sequence of wizard windows using the Script Wizard, a script developer identifies requirements for the script, and specific panels, questions, and answer choices within the script.

Wizard scripts result in scripts identical in composition to graphical scripts, with two exceptions: you cannot open a graphical script using the Script Wizard, and wizard scripts do not contain all the objects accessible to the graphical script user interface.

runtime

Runtime is the execution of any script created using Script Author. Scripts are executed

using the Scripting Engine component, which has two interfaces: the agent interface and the Web interface.

The agent interface executes as a Java window launched from an Oracle business application form. The agent interface displays the panel presentation area, a script information area, a shortcut button area, and a progress area, wrapped in a frame with a Disconnect button and, optionally, a Suspend button.

The Web interface executes a script in a Web browser. The Web interface displays the panel presentation area, and the following buttons: Back, Next, Reset to Default, and optionally Save for Later.

Before any script can be executed in the Web interface, survey administration is required, which includes setting up survey resources, creating a hierarchical set of objects - a survey campaign, cycle, and deployment - and activating the deployment.

Script Author Concepts and Features

This section includes the following topics:

Scripts

A script is a miniature program built using Script Author by a trained functional user (a script developer).

The purpose of a script is to facilitate the flow of information at runtime between an enterprise and other parties. Each script enforces business rules programmed into it by script developers.

Scripts consist primarily of panels, which display at runtime. The composition of each panel is determined by script developers using Script Author, but must contain at minimum a single question. Panels may include any number of questions (there are no limits other than practicality).

Depending on the user interface at run time, users progress through the script, typically by clicking a Next or Continue button.

Although a rigid flow can be enforced, effective scripts typically include carefully constructed logic that branches the end user through different potential paths to appropriate panels, based on responses to earlier script questions. Answers provided at runtime can be changed by the end user before the script is completed. If an answer is changed by the script end user, and the logic for progressing through the script is programmed to change accordingly, the new path will automatically result. If an end user backs up in a script (for example, from panel 10 to panel 5), and changes an answer, and if the answer does not affect branching, then after registering the changed answer (by clicking Continue in panel 5), the script flow resumes at the last unanswered panel (panel 10). In this way, unnecessary duplication of responses is programmatically avoided. This is the intentional behavior of the application, which cannot be changed.

Script Author provides the ability to create two kinds of scripts:

- Graphical Scripts, page 2-4
- Wizard Scripts, page 2-19

Regardless of creation method, a script can be executed in any interface of the Scripting Engine.

Graphical Scripts

A graphical script is any script created or modified using the graphical tools of Script Author. Graphical scripts employ a visual paradigm similar to a flowchart. Using drag and drop techniques, script developers place graphical objects in a work area (the canvas), and associate properties to those objects. This provides an intuitive method to build, modify, examine, and extend the functionality of a script.

Graphical scripts can contain configurable objects (panels, groups, and blocks) and nonconfigurable objects (start and termination nodes). The main features of each these script objects are the following:

- Panels are the only script objects to display at runtime. Each contains at least one question , and (in a graphical script) may contain text, graphics, hyperlinks, and embedded values.
- Groups are subgraphs which can logically organize related functions.
- Blocks are subgraphs which provide a hook for APIs or SQL functions.
- Start and termination nodes are nonconfigurable because the only function they serve is to signal the beginning or end of processing for a graph in a script. A termination node on the root graph (the first graph that appears when a script is opened) represents the end of script processing for the entire script.

Objects in a graphical script are connected with various branch types to control flow at runtime: default branches, distinct branches, conditional branches and indeterminate branches.

Graphical Script Components

Scripts consist of the following:

- Configurable objects, page 2-5
- Nonconfigurable objects, page 2-7

Objects are processing units that tell the script what to do.

Objects are connected by branches, page 2-8. Branches are processing units that tell the script where to go.

A graphical script contains, at minimum, one graph. This is called the root graph - the graph which is visible when you open any graphical script.

Two objects of a script, groups and blocks, serve specific functions, but do not display at runtime. Groups and blocks serve as containers for other graphs, which are referred to as subgraphs.

When the script is executed, it displays panel content (which may include text, prompts, hypertext links, embedded values, and images) to its end users. Each window is the graphic display of a panel object, page 2-6. In each panel, the end user must provide a response - a mouse click or keyboard command - before progressing in the flow to the next object in the script.

Configurable Objects

There are three *configurable* objects in Script Author. A configurable object is an object which has properties that you can modify.

The three configurable objects in a graphical script are:

- Panels, page 2-6
- Groups, page 2-6
- Blocks, page 2-7

Panels

Scripts consist primarily of panels, the only script objects to display at runtime. The composition of each panel is determined by script developers using Script Author, but must contain at minimum a single question. Panels may include any number of questions (there are no limits other than practicality). Panels in graphical scripts may include text, graphics, hyperlinks, and embedded values. Panels in wizard scripts may contain text in addition to any number of questions.

At runtime, each panel displayed in the agent interface includes a Continue button. In the Web interface, the frame surrounding the panel includes a Next button. Runtime script end users must click the Continue or Next button to progress through the script.

In a graphical script, panel layout can be customized on a panel-by-panel basis. Panels containing graphics must be customized to refer to the graphic on a Web server accessible to the Scripting Engine at runtime.

Groups

A group is a container for other objects and branches. It may contain any type of object, including other groups. Groups can be nested as many levels as desired (the only limit is practicality). A group represents a child graph (a subgraph); as such, the graph represented by the group object must meet certain branching and termination requirements (see Minimum Requirements For Any Graph, page 2-27).

Groups are used to:

- Logically group a section of the script flow (typically, to contain a single function or process)
- Serve as a container for a Shortcut. In the Scripting Engine agent interface, shortcuts appear in the shortcut button area. When clicked at runtime, the agent user is "jumped" to the group in the script containing the referenced shortcut.
- Represent an imported script in a graphical script .

Disconnect Group

Some enterprises require the collection of extra information for every script. In a graphical script, if you put the objects collecting this information in the Disconnect group, you can ensure this information is solicited for every interaction. If you add panels to the default Disconnect group, Oracle recommends that you change its name (for example, to WrapUpGroup), to provide a visual indicator to script developers that the group includes content for the wrap-up of each script transaction.

By default, wizard scripts contain a single group called Disconnect. The Disconnect group in the wizard script contains no panels, groups or blocks. It simply represents a graph with start and termination nodes connected with default branching, and serves to end the script immediately upon invocation.

This group contains the shortcut property WrapUpShortcut. When the Disconnect button is clicked at runtime from the Scripting Engine agent interface, the group in the script (if any) containing this shortcut is the resulting destination of the script.

> **Note:** The Disconnect button always appears in the agent interface. If users click the Disconnect button, and WrapUpShortcut or its equivalent is not defined for the Disconnect group, they will get an error.

Blocks

A block is used for one of two purposes:

- When used to query, update, or insert information in a database, it is always associated with one or more database tables and contains database connection information.
- When used as a container for an Application Program Interface (API), it contains a Script Author command that references the API.

A block represents a child graph, also known as a subgraph. The subgraph represented by the block object must meet the standard branching and termination requirements (see Minimum Requirements For Any Graph, page 2-27).

The block subgraph can contain panels. The question UI controls in the panel objects can be used in block processing. The subgraph panels are executed *before* the main query, insert, update, or API operations associated with the block.

Nonconfigurable Objects

There are two *nonconfigurable* objects in Script Author. A nonconfigurable object is an object which has a single function, and does not contain properties that you can modify. The two nonconfigurable objects in a graphical script are start and termination nodes.

Start Nodes

Every time a graph is created in Script Author, it contains a start node. Start nodes cannot be explicitly created, nor deleted. They contain no viewable properties. Start nodes simply visually represent the starting point on any graph. They can be moved about the canvas as desired, and must be attached using a default branch to the first explicitly created object in the script. Any actions associated with a branch from a start node will be ignored.

Termination Nodes

Termination nodes are required on every graph (unless that graph contains an indeterminate branch). Termination nodes represent the end of the flow of that level of the script at runtime. They are nonconfigurable, containing no viewable properties. A graph may contain two or more termination nodes, each depicting the end of a flow of a particular path. The termination on the root graph signifies the end of script processing.

Branches

Objects in a graphical script are connected with various branch types to control flow at runtime. There are four branch types:

- Default: The destination is the designated object.

  If there is more than one branch from an object, this type of branch must be defined as the last-case default if no other branch is valid. Requires no properties to be associated with it in Script Author. There is no label associated with this branch type, and its name is not required to be unique within a script or even on one graph of a script.

- Distinct: The destination is based on the answer provided to a question at runtime in a panel. One or more distinct branches are designed to start from the panel.

  The Name property of the distinct branch appears as a branch label on the canvas. This value is not required to be unique.

  The Value property of the distinct branch determines the path the script will take if the specified value is selected as the answer choice to a panel question by the script end user at runtime.

  You must define one or more answer choices (or lookup values) for a single question - designated as the default for distinct branching - in the panel. For each distinct branch drawn on the canvas from that panel, you must associate one of the answer choices, to specify the correct flow at runtime when the script end user answers the designated question.

- Conditional: The destination is based on the evaluation of a Boolean expression.

  For this branch type, you must provide and associate Java code to the conditional branch that evaluates a condition to be true or false. The branch is taken at runtime if the Boolean expression is determined to be true.

  > **Note:** When you define one or more conditional branches for an object, you must also provide a default branch, to allow for the case that all the conditions are false. Because branches are evaluated in the order specified for the object, ensure that all conditional branches for an object appear before the default branch - see

Branches and Branch Order, page 2-9.

- Indeterminate: The destination is based on the evaluation of an expression, the result of which is not known until runtime.

  This branch type requires the definition of a Script Author Java command to be associated with the branch in the script, referencing corresponding Java code (to provide the expression and destination objects) to be evaluated at runtime. The expression must return a string that is either the name of a sibling object (panel, block or group on the same canvas), or the shortcut name of a destination group (using the Shortcut property), which can be in any graph at any level of the script from the root graph to the deepest nested level.

  > **Note:** The terms *indeterminate* and *indeterminant* are used interchangeably in Oracle Scripting.

Branches and Branch Order

Below the start node, each object placed on the Script Author canvas in a graphical script is typically connected above and below by appropriate branches, ending with the termination node. Outgoing branches (branches drawn *from* any object) are displayed in the Branch pane that results by clicking the Branch attribute under any object's Property window.

The branches are displayed in the order in which they are created. By default, this determines the order in which the branches are evaluated at runtime. You can change this order after branches are created. Thus, for a panel with multiple branch types, it is crucial to ensure the Default branch is listed last so that other branch types will be evaluated first.

Panel Layout Editor

The panel layout editor feature of the Script Author allows script developers to quickly define simple HTML layouts for a panel within a graphical script without writing HTML code. This simple graphic interface provides users with the ability to:

- Enter and format simple panel content
- Manipulate typeface, font size and color
- Create bulleted and ordered lists
- Import existing HTML or export panel content as HTML
- Create hypertext links

- Insert embedded values
- Import GIF or JPG images

> **Note:** When an image is imported using the panel layout editor, the HTML for the panel must be exported from the panel in a graphical script, and modified to reference the image in a path accessible to the Oracle Applications Web server at runtime. The modified code is then reimported into the panel, so the image is available to all script end users at runtime.

In addition, the panel layout editor produces syntactically correct HTML content for Scripting-specific features such as embedded values without script designers having to learn custom Oracle Scripting HTML tags and syntax.

Questions (or question UI controls), images, and text can be interspersed and formatted as desired.

The panel layout editor includes one-click formatting of text into two distinct styles:

- *Instructionaltext* indicates specific instructions to the script end user.
- *Spoken text* indicates text that an agent would speak (using the agent interface) or that a customer or survey respondent would read for communication of primary information for that panel.

The panel layout editor is not intended to be used for complex layout, creation of use of HTML tables, or the use of cascading style sheets, which are not supported with Oracle Scripting. To create complex layout, or to use features of HTML such as tables, it is expected that script developers will use full-function third-party HTML editors (or code HTML by hand) and simply import the HTML content into Script Author using the panel layout editor.

Questions

> **Important:** In Oracle Scripting, there is a one-to-one relationship between questions and answers.
>
> When describing Oracle Scripting questions and answers, the terms "question" and "answer" are used interchangeably, with the more appropriate term selected in context.

Every panel requires at least one question user interface (UI) control, because at runtime, every panel requires end user interaction (a mouse click or keyboard action) in order to progress to the next object in a script

Note: When defining a question UI control, you must select Answers from the panel properties tree. The window in which question UI controls are defined is referred to as the Answer Entry Dialog window.

The terms "Question UI Control" and "Answer" are used interchangeably in this section.

For each question UI control you specify, the panel will include one question at runtime.

The question UI control types and their general usage are as follows:

Question UI Control Type	Description
Button	Use to progress the script to a panel. *Cannot be defined in Script Wizard..*
Check Box	Use to evaluate a simple "true/false" condition. *Cannot be defined in Script Wizard.*
Checkbox Group	Use for a series of conditions for which you can select zero, one, or many values at runtime.
Drop Down	Use for a series of conditions for which you must select one value at runtime.
Multi-Select List Box	Use for a series of conditions for which you can select zero, one, or many values at runtime.
Password	Use for short amounts of text to be entered, with entered data appearing as asterisks.
Radio Button	Use for a series of conditions for which you must select one value at runtime.
Text Area	Use when soliciting large amounts of text at runtime, with space for inputting several lines.
Text Field	Use when soliciting a short text response at runtime.

General Question Categories

Data Entry or Data Selection

The following question UI controls require users to enter data at runtime: Text Field, Text Area, and Password. All the other questions require users to select one or more values at runtime.

Single-Value or Multi-Value Selection

Radio Button, Check Box, Button, and Drop Down questions require a single value to be selected.

Checkbox Group and Multi-Select List questions allow for zero, one or many values to be selected.

Question Properties

Each question, whether defined in a graphical or wizard script, has a number of question properties, such as whether it requires a label or a lookup values.

Each question in a panel has a data dictionary associated with it, which stores the main question properties. For example, for questions that support multiple answer choices (radio buttons, drop-down lists, checkbox groups, and multi-select list boxes), these answer choices are determined by the contents of the question's data dictionary.

In wizard scripts, the question properties are visible in the standard question and answer choice windows. In graphical scripts, you can view and edit question properties explicitly in a data dictionary window.

How Many Questions Are Required Per Panel?

A panel must contain at least one question, and may include any number of questions. There are only two limits: practicality and distinct branching. From a practical standpoint, at runtime, the user should be able to easily navigate each panel. The more questions defined in a panel, then the more extensive the panel layout. Users generally prefer to view the contents of a panel in a single screen, when possible.

On this point, scripts differ in philosophy based on their intended use. When a script is created to collect survey data (whether executed in the agent interface, or more commonly, when executed in a Web browser), it is generally considered acceptable or commonplace to include a large number of related questions on a single page. Often, these are questions of the same type and share the same attributes. For example, you may include a dozen questions asking about satisfaction of a certain aspect of a product or service, each using radio buttons and requiring the end user to provide a rating from one to five.

Regarding distinct branching, only one question per panel can directly affect distinct branching. If a panel uses distinct branching for an outgoing branch type, the question that uses distinct branching *must* have the "Default for Distinct Branching" option selected.

Shortcuts

A shortcut is a property of a group in Script Author. The shortcut exposes the functionality of the group and its contents to the Scripting Engine, enabling that group to be the target or destination of a "jump" from another location of the script.

This jump occurs when the shortcut is invoked using the Oracle Scripting jumpToShortcut API. This API is associated with the script using a Java command, which can be called as an action, pre-action or post-action anywhere in a script.

For more information about commands and actions, see Defining Commands, page 8-89 and Defining Actions, page 8-88.

The shortcut property of a group is also used by the indeterminate branch. This branch type requires an expression to be defined. This expression is a Java method which, when evaluated, provides a shortcut name as its return value (based on the conditions tested in the method), invoking the shortcut.

Regardless of whether it is called from an action or an indeterminate branch, when the shortcut is invoked, the flow of the script at runtime jumps from the existing object being processed to the group containing the shortcut name that is returned by the command.

At that time, any pre-actions associated with the group are executed. The group represents a subgraph. Thus, processing then focuses on the objects within the group. When the termination node within that group is reached in the script flow, then any post-actions associated with the group are executed. Flow then returns to the graph on which the shortcut group resides, directed by the outgoing branch of the group containing the shortcut.

If the group containing the shortcut contains one or more panels, then the effect to the script end user after a shortcut is invoked is simply that the script progresses to the first panel within the target group. Processing of any commands, of course, is hidden from view, although data returned from such processing can be displayed at any point during that script session, and commands relying upon such data can be executed at any point during that script session.

Since only panels display at runtime, regardless of which Scripting Engine interface is used, then if the group containing the shortcut contains no panels, then the flow of the script is redirected to a new path. That path starts from the target group. Group or block objects within that group (if any) and any commands associated with them (if any) are processed. At minimum, the group containing the shortcut that was called contains a termination node. When that termination node is reached in the script flow, then processing and script flow returns to the graph on which the shortcut group resides. The script has effectively been redirected to whatever path follows from the shortcut group. In this case, the visual effect to the script end user after the shortcut is invoked is to see the first panel in the new flow of the script, starting with the target group. Again, all processing from other objects or command occurs behind the scenes.

Typical uses of a shortcut are as follows:

- To associate a target for a shortcut button in the shortcut button area (displays for the agent interface only).
- To associate a target for a group containing specific functionality which must be accessed after meeting a specified condition in a script (for either runtime interface).
- To redirect the flow of a script to a new path (starting with the group that contains the shortcut).

Global Script Properties

Global script properties are those properties that apply to an entire script. This is in contrast to properties of a specific object in a graphical script such as a panel, group, block, or branch. Global script properties are assigned default values when a script is created, and exist for wizard scripts and graphical scripts alike.

Global script properties fall into four categories:

1. General global script properties
2. Actions
3. Static panel
4. ld

Details on each of these properties appears in the table below.

Global Script Property	Graphical Script UI Type	Wizard Script UI Type	Graphical Script Default Value	Wizard Script Default Value
Script Type	N/A	N/A	Graphical script	Wizard script
Name	Text field	Text field	Untitled1	Untitled
Comments	Text field	N/A	Null	N/A
Description	N/A	Text area	N/A	Null
Script Language	Drop-down list	Drop-down list	AMERICAN	AMERICAN
Footprinting	Boolean	N/A	TRUE	TRUE

Global Script Property	Graphical Script UI Type	Wizard Script UI Type	Graphical Script Default Value	Wizard Script Default Value
Answer Collection	Boolean	N/A	TRUE	TRUE
Suspendable	Boolean	N/A	TRUE	TRUE
Override suppression of Continue button	Boolean	N/A	FALSE	FALSE
Pre-Actions	Null	N/A	Null	N/A
Post-Actions	Null	N/A	Null	N/A
Static Panel	Null	N/A	Null	N/A
Shortcut Panel	Null	N/A	Null	N/A

The Static Panel and Shortcut Panel global script properties only affect the Scripting Engine agent interface. These properties, when appropriately configured by a script developer, cause additional objects to appear in the script frame. Like the progress area, the Static Panel and Shortcut Panel properties (if used) are rendered as Java beans that appear in the designated areas of the script frame of the agent interface only. Scripts executed in a Web browser do not display any objects in the script frame other than the panel display area.

Graphical script users can disable the Boolean general script properties for footprinting, answer collection, and script suspendability in the Properties pane of the global script properties window. Accessing global script properties from a graphical script is the *only* method to disable Boolean general script properties or to add actions, static panel or shortcut button area information.

Script Type

The first property, Script Type, is determined when you select Graphical script or Wizard script in Script Author (File > New). The sole purpose for this property is to distinguish whether the script is a wizard script or graphical script. The Script Type property is not modifiable, nor is it visible from any Oracle Scripting user interface. Nevertheless, this property is determined by the method you use to create a script.

Script Name

The script name is the name by which the script is identified in the database. This is

distinguished from the file name of the script (which must contain a file extension of `.SCR` or `.SCRIPT`). Avoid using special characters such as slash or backslash, percent, asterisk, or other characters that have special meanings to a filing system.

Comments or Description

The comments (graphical script) or description (wizard script) fields allow script developers to enter or view comments or descriptions regarding the particular script.

Script Language

This property is a drop-down list which contains the value of the appropriate language for your environment from the Oracle RDBMS FND_LANGUAGES table. The default value is AMERICAN.

Footprinting

Footprinting records the sequence of panels enabled during a script runtime interaction (regardless of whether the script is viewed in the Scripting Engine or the Survey mode), as well as the start time and end time (in milliseconds) for each panel in an interaction.

Additionally, footprinting records deleted status (indicating that a panel was removed from the final flow based on a changed answer that takes the user down a different flow path).

Footprinting data provides interaction center managers the ability to review existing scripts to determine potential problems and to tune the script accordingly. For example, you can analyze footprinting data to identify areas in the script where substantial amount of time is spent or where a specific panel is often deleted during an interaction. These scripts can then be modified with the goal of decreasing average talk time (for an interaction center) or increasing valid response rate by improving the flow of a script.

The default for this property is true.

When enabled, this data is saved in tables IES_PANEL_DATA and IES_FOOTPRINTING_DATA in the Oracle Applications schema.

Answer Collection

Answer collection is the recording of end user responses ("answers") to all question user interface (UI) controls or "questions" during a script transaction.

The default for this property is true.

When answer collection is enabled, script end user responses are collected for each question designated as collectable, and saved in the IES_QUESTION_DATA table in the Oracle Applications schema.

For graphical scripts, individual questions can be designated as uncollectable.

Suspendable

The Suspendable check box option allows scripts to be suspended and resumed, in both of the Scripting Engine interfaces.

The default for this property is true, for both graphical and wizard scripts. The property is visible and editable only from a graphical script, in the script properties dialog (File > Script Properties). The only way to disable the Suspendable option for a wizard script is to graph the script and then edit the graphical script.

The effect of this option at runtime depends on the user interface:

1. For suspendable scripts running in the agent interface, if the profile option IES : Display Suspend Button on Script Frame is set to True, the agent interface displays a Suspend button on the script frame. This button is hidden if the profile is set to False (or remains null).

2. For deployments running in the Web interface, when the survey campaign associated with the questionnaire script is marked as Suspendable (either Always Suspend or Allow Suspend), a Save for Later button appears in the Web interface at runtime. The appearance of the button does not depend on any profile option value.

When you click either the Suspend or the Save for Later button at runtime, the current script interaction is suspended.

The way to resume a suspended script also depends on the user interface.

Scripts suspended in the agent interface can only be resumed through facilities built into the Oracle application that launched the script; currently only Oracle TeleSales provides this capability.

> **Note:** For testing purposes only, you can resume a script suspended in the agent interface from the Navigator by an Oracle Applications user with an appropriate responsibility to access Oracle Scripting scripts.

The resumption of scripts suspended in the Web interface depends on whether the scripts executed from a standard or a targeted deployment.

A resumed transaction retains all information collected in a script up to the point at which it was suspended, including footprinting and answer collection information.

Override Suppression of Continue button

When a script is executed in the Scripting Engine Web interface, by default, any Continue buttons created in the script panels are suppressed. This feature is available because the Web interface includes Next and Back buttons to enable you to progress through the script.

If you wish, you can choose to override this suppression, and the Continue buttons will then be visible in the panels.

You cannot suppress the appearance of Continue buttons in scripts executed in the agent interface.

Pre-Actions and Post-Actions

Global script actions (pre-actions and post-actions) are commands that execute prior to the display of the first panel in a script sequence or after the last panel is displayed in a script sequence, respectively. These global script properties can only be assigned from graphical scripts.

Static Panel

The Static Panel, also known as the script information area, is an area that displays, in the agent interface only, above each panel, as the script is executed.

The script information area can contain up to nine data elements. The elements can be of two types, text and timer.

Text elements are for static information, such as a campaign name or business purpose of the script. You provide the static information by associating a text element with a command that specifies either a constant value or a value retrieved from the Oracle Scripting blackboard.

> **Note:** At the start of script execution in the agent interface, you can store blackboard values and parameters either from an Oracle application that calls the script or from a script pre-action.

> **Note:** The static panel text elements are populated after any script pre-action, and cannot be subsequently changed.

Timer elements are typically used to display dynamic information, from dynamic timers. Using Scripting APIs, timers can be started, stopped, and resumed; and data in the script information area can be updated by explicit command.

You provide timer information through a Java method that invokes the Oracle Scripting API startTimer. You can do this in several ways:

- Associate the Java command with a shortcut button, which when clicked at runtime, will start the timer.
- Attach the Java command as a pre- or post-action command for a script or script object.

Shortcut Panel

In the agent interface at runtime, below the script information area and immediately above the panel display area, the shortcut button area displays shortcut buttons, which may be associated with any Script Author command. You define the shortcut buttons in

the Shortcut Panel.

Shortcut buttons contain a command that is executed when the button is clicked in the agent interface at runtime. Often, shortcut buttons contain a command referencing the Oracle Scripting jumpToShortcut API. In this case, the command returns a destination panel or group in the script. When clicked, the script flow at runtime "jumps" to the specified panel (or the first panel in the specified group).

If the button contains another command, the command is executed on the click action. Popular uses for these buttons are to open a browser window with a specified Web address, or to enable or disable an interaction timer in the script information area.

Shortcut buttons can be enabled or disabled at the start of a script. When a shortcut button is enabled, clicking on it at runtime will perform the function associated with the button. When disabled, clicking the button will have no effect. The state of the shortcut button is set when you define a shortcut button, and can also be controlled by invoking Oracle Scripting shortcut button APIs.

Hierarchical Graphs

A script can contain any number of graphs. The graph shown when you first create or open a script is referred to as the *root graph*. Any new graph (subordinate in hierarchy to the root graph), created at any level of a script, is known as a *child graph* or *subgraph*. The graph from which a subgraph is created is known as the *parent graph*. The root graph has no parent.

New graphs are created using group or block objects. After creating a group or block object in a graphical script, you must explicitly select it and navigate down into it to view the child graph. Any graph can contain any number of other Script Author objects (including any number of additional child graphs, if required).

When an object in the flow of the script contains a child graph in the form of a group or block, any objects included in that child graph are evaluated before the flow on the root graph continues to be evaluated. A child graph at any level is exited at runtime when the termination node on that graph is reached in the flow of the script. Processing (flow of the script) then returns to the next object in the parent graph.

Certain syntactical rules apply to every graph. All objects in a script must be connected with branches. At minimum, each graph must contain a start node and a termination node, connected by a default branch. When you include a group or block in a script, its child graph must meet the same syntactical rules as the root graph.

Wizard Scripts

Script Author includes a Script Wizard feature. Using this feature, script developers can quickly and easily create simple scripts or surveys by providing script information in a series of windows known as a wizard.

Scripts created with the Script Wizard are called wizard scripts. All features of wizard

scripts are compatible with those of graphical scripts, and wizard scripts can be graphed or converted to a copy of a graphical script for viewing or modification using standard graphical tools. There is no backward compatibility, nor can graphical scripts be converted to wizard scripts.

The Script Wizard makes script development more accessible to non-technical users, although an understanding of business rules and business process flow is still absolutely essential to developing successful scripts using this feature. Since wizard scripts are created from Script Author, access to the Scripting Administrator responsibility is also required.

Wizard scripts can be created, edited, saved to the database without deployment, or deployed to the database. Unlike graphical scripts, wizard scripts cannot be saved locally in a computer's file system. Wizard scripts can only be listed in a user interface from within the Script Wizard.

Highlights of wizard scripts include the following:

- A wizard script cannot be saved or deployed until it is syntactically correct. When you complete a wizard script, you can save it and continue working in the wizard, you can save and exit the script, or you can save, deploy, and exit. The product of progressing through the Script Wizard and saving your work is always a valid, deployable wizard script.

- From the Script Wizard, existing wizard scripts can be edited, copied, deleted, graphed for use in the graphical Script Author tools, or deployed.

- Each wizard script automatically contains a valid Disconnect button. This is only visible in the Scripting Engine agent interface.

- Each wizard script automatically contains a valid Suspend button, unless the Suspend feature is disabled through other means. This is only visible in the Scripting Engine agent interface.

- When defining question detail, wizard script developers can include default answers to questions by providing a value in the Default Value field. These can be accepted or changed at runtime.

- When defining question detail, using a few clicks, wizard script developers can include answer validation for responses provided at runtime. You can check that the response provided is either (a) an integer, (b) an integer within a designated start and end range, (c) a valid date, or (d) a date that is not in the past (e.g, the date is equal to SYSDATE or is in the future).

- Flow of a script can be controlled in part by providing an exit panel sequence. This method allows panels meant to contain information only (in contrast to panels designated to collect information) to automatically be provided with a Continue button without requiring the Script Wizard user to specify a panel answer.

Wizard scripts can contain panels, default branches, distinct branches, start and termination nodes, and a Disconnect group. Panels contain questions which may have predefined answer choices (lookup values).

For advanced wizard script features, any question may contain a default answer. Additionally, questions requiring text input may have the aforementioned answer validation. Corresponding features in graphical scripts are provided, respectively, by defining constant commands or Java commands referencing custom or best practice Java methods. These are defined for graphical scripts in the data dictionary for the particular question.

Comparisons Between Graphical and Wizard Scripts

Graphical Scripts

Branching and Graphical Scripts

Decision-enabling data for branching graphical scripts includes previous end user responses for the current session, information received from external database tables or applications through forms commands, blackboard commands or PL/SQL commands, and events or conditions such as the time of day.

Advantages of Graphical Scripts

- Substantial control of the appearance of a panel at runtime
- Ability to add panels to the WrapUpShortcut Group
- Ability to change the display value of a Continue button
- Ability to associate commands with script objects
- Ability to provide conditional and indeterminate branching
- Ability to gather related functionalities into distinct groups
- Ability to disable Boolean global script properties for footprinting, answer collection and script suspendability
- Ability to include graphics in a panel
- Graphical scripts intended for execution in the Scripting Engine agent interface may contain global script properties that enable you to define create shortcut buttons and a script information area, which display at runtime in the Scripting Engine agent interface.
- In a graphical script you can specify an action as a Script Author command, and

associate that action with any object in the script (including the global script object itself). Objects accept pre-actions and post-actions, which are executed either before or after that object is processed at runtime. Branches include actions, which occur as that branch is processed at runtime. By comparison, the only actions that can be associated with wizard scripts are seeded best practice Java methods to provide answer validation to defined questions, or constant commands to provide default answer choices.

Disadvantages of Graphical Scripts

- Must manually create a WrapUpShortcut Group
- Must reference Best Practice Java methods by specific class and method name, including for answer validation
- Must define Continue button as panel question
- Many steps required to accomplish simple tasks
- Scripts can be saved in syntactically incorrect state
- Any graphics in a panel require panel HTML customization to reference the graphic on a Web server so that other users can view the graphic at runtime

Wizard Scripts

Branching and Wizard Scripts

Decision-enabling data for branching wizard scripts is programmed into a wizard script at two junctures: when specifying a panel's exit panel sequence, and (for those question types that accept answer choices) when defining specific answer choices. Thus, at runtime, branching is controlled at the panel level generally. Wizard script panels with answer choices can define overriding branching instructions. For these panels, in the event that one specific answer choice for one specific question in the panel is selected, a new distinct path is taken for that session of the script.

Wizard scripts cannot be assigned commands other than for panel validation. Thus, wizard scripts do not branch based on the evaluation of commands, and are *not* routed based on information received from external database tables or applications through forms commands, blackboard commands or PL/SQL commands, or events or conditions such as the time of day.

Advantages of Wizard Scripts

- Saved scripts are always syntactically correct by default.
- Each script automatically includes a valid WrapUpShortcut group as the runtime

destination of the Disconnect button.

- Wizard script users can employ seeded Java methods to provide answer validation within a panel, without requiring a single line of code or without even the need to specify the specific Java class or method.

  To do so, users of the wizard specify, in the Define Question Detail wizard window, whether a response is mandatory, if a default value for the question response should appear at runtime, or the type of validation.

- Wizard script users also do not need to provide answer choices when the primary purpose for a panel is to provide an informational message and then continue to the next panel.

  By specifying "Go to the next panel in sequence" as the exit panel sequence, a Continue button will automatically be generated by the Script Wizard. In contrast, graphical script users must access panel questions, define a question name and question UI type of Button, and access the data dictionary for the question to provide a display value and passed value of the word "Continue."

- Wizard scripts can be saved in the database, or deployed to the database for execution, directly from wizard windows. This feature can help reduce the time it takes to train a non-technical Script Author user to create and modify simple scripts or surveys.

- Wizard scripts can be *graphed*, a process in which a copy is converted to a graphical script. Since this process is not reversible, the original wizard script is retained in the database for future access using the wizard. The ability to graph a wizard script allows graphical script users the major benefits of the Script Wizard (rapid script creation, keystroke reduction, and elimination of repetitious processes) by allowing them to begin a script using the wizard, and to subsequently have access to the more complex functionality accessible only in a graphical script.

Disadvantages of Wizard Scripts

- Once wizard scripts are graphed, this process is irreversible. Any enhancements or changes made to a graphed script must be accessed as a graphical script only.

- You cannot associate any commands to scripts globally, or to specific objects in a script, using the wizard.

- You cannot disable the Boolean global script properties (footprinting, answer collection, or script suspendability) from a wizard script. The script must be graphed so that these options can be disabled.

- You cannot include pre-actions or post-actions to the global script.

- You cannot include the script information area or shortcut button area in wizard

scripts. These global script properties (which display at runtime only when the script is executed using the Scripting Engine agent interface) are only configurable in a graphical script.

- You cannot create conditional or indeterminate branching using the Script Wizard.
- You cannot automatically include panel layout text in more than one format (for example, spoken text and instructional text) using the wizard.
- You cannot create embedded values in panel text using the Script Wizard.
- You cannot create hyperlinks or add graphics to panel layouts using the Script Wizard.

Differences Between Wizard and Graphical Scripts

Wizard scripts and graphical scripts are both created from Script Author, and both can be executed in any Scripting Engine interface. But there are many differences, some of which are identified here.

Separate File Structures

Using the Script Wizard results in scripts identical in composition to graphical scripts, but with separate file structures. These present some practical limitations.

- You cannot open a graphical script using the Script Wizard.
- From the Script Wizard, you can graph a wizard script, allowing you to see the resulting physical layout of a script you created using the wizard tool.
- Any changes you make to the graphical copy cannot be viewed or edited from the Script Wizard.

Wizard Includes Limitations on Global Script Properties

Global Script Properties fall into four categories: General global script properties, Actions, Static panel, and Shortcut panel. Of these, only general global script properties can be configured for wizard scripts. Actions, Static panel properties (for the Scripting Information window in the agent interface), and Shortcut panel properties (for shortcut buttons in the agent interface) are only configurable in graphical scripts.

Boolean Property Limitations in Wizard Scripts - Wizard scripts limit access to some of the general global script properties. For example, there are three Boolean properties (Footprinting, Answer Collection, and Suspendable). These are automatically enabled for all wizard scripts.

Graphical script users can disable the Boolean general script properties for footprinting, answer collection, and script suspendability in the Properties pane of the global script properties window. Accessing global script properties from a graphical script is the

only method to disable Boolean general script properties or to add actions, static panel or shortcut button area information. A wizard script must be graphed before any of these properties can be modified.

Comments or Descriptions - Graphical scripts include a Comments text field as a global script property. Wizard scripts include a Description text area as a global script property. Text entered in either of these fields are not saved to any columns in the database. They are stored in the .SCRIPT file and are not displayed anywhere at runtime, nor are they visible from the Scripting Administration console. The sole purpose for these fields is to allow script developers to enter or view comments or descriptions regarding the particular script, and to allow those comments to be viewed by other script developers.

Different Script Types

The Script Type is a global script property. This property is not viewable or modifiable from any Oracle Scripting user interface. However, this property (the value for which is either wizard script or graphical script) identifies the script creation method and thereby, the type of script. When you graph a wizard script, you create a graphical copy (with a graphical script type); the original wizard script is retained.

Panel Objects Included in Wizard Scripts

- Panels created with the Script Wizard do not include the single checkbox or the button (submit button or push button) question user interface controls.
- Script wizard panels also cannot contain graphic images, hyperlinks, or embedded values.

  To add any of the preceding to a wizard script, or to add groups (other than Disconnect) or blocks, you can convert a wizard script to a graphical script. *Any modifications you make to the graphed script cannot subsequently be viewed using the Script Wizard tool.*

Question UI Control Differences

The most obvious difference between panels created with the Script Wizard and in a graphical script is the limitation of question user interface types. The table below compares the question UI types available to graphical scripts versus those accessible in the Script Wizard. Minor differences in control names are indicated based on the labels provided by the user interface for each script creation method.

Graphical Script	Wizard Script	End User Action Required
Text Field	Text	Text entry
Text Area	Text Area	Text entry

Graphical Script	Wizard Script	End User Action Required
Radio Button	Radio Button	Select only one (required)
Check Box	[None]	Boolean (null or selected)
Button	[None]	Click (required)
Drop Down	Dropdown	Select only one (required)
Password	Password	Text entry
Checkbox Group	Checkbox Group	Select zero, one, or many
Multi-Select List Box	Multiselect List	Select zero, one, or many

Panel Properties Differ Based on Creation Method

Panels in wizard or graphical scripts may contain static panel text, and one or more questions. Questions of the appropriate type that are explicitly defined in a wizard script may include question-level validation commands. In addition, panels in graphical scripts may contain graphic images, dynamic text using embedded values, and validation, database or cursor lookup, or other commands (at the panel level or the question level).

Explicit or Implicit Question Definition

If the purpose of a wizard script panel is simply to provide information to the script end user, then Script Wizard users are not required to define any questions for the panel. The Script Wizard will automatically generate a question called Continue, which displays a Continue button at runtime.

In contrast, panels created with graphical tools *must* contain one or more explicitly defined questions. If the question UI control selected for a panel is Button, the panel may not contain any other questions. A display value (the contents of the button) and a value must be provided for each button question type created, in the Lookup entry window of the question data dictionary.

Although identical in appearance to the Button question UI type, panels in graphical scripts containing one or more question UI types other than Button will automatically display a Continue button at script runtime. This button is not required to be defined.

Objects and Branch Types Included in Wizard Scripts

- Wizard scripts contain panels, branches, start and termination nodes, and a single group (Disconnect) to enable the Scripting Engine agent interface Disconnect

button. Wizard scripts do not include any other groups or any blocks.

- Wizard scripts do not include conditional or indeterminate branching.

  To add groups or blocks, conditional or indeterminate branching, or to modify the Disconnect group, you can convert a wizard script to a graphical script. *Any modifications you make to the graphed script cannot subsequently be viewed using the Script Wizard tool.*

Explicit Question Definition for Graphical Scripts

Each graphical script contains at least one explicitly defined question, resulting in a user interface control displayed in the panel at runtime. If no other question user interface controls are required for the panel, developers of graphical scripts must explicitly define the Continue button, using the Button question UI control. Panels with other question types will include an automatically generated Continue button.

In contrast, *each* panel in a wizard script includes an automatically generated Continue button. If no other question user interface controls are required for the panel, then wizard scripts differ from graphical scripts by generating the button automatically. You cannot choose to create a Button question UI type, nor can you provide the default button with any value other than Continue from the Script Wizard.

Minimum Requirements for Any Graph

A graphical script is represented as one or more graphs that use visual symbols to represent the three configurable object types (panels, groups, and blocks) connected with the appropriate branch type to control the flow of the script.

Each valid graph within a script will also contain a start node (created on each graph or subgraph automatically) and at least one termination node (inserted on the canvas by the script developer). These two nonconfigurable objects will also be connected with branching as appropriate.

The Shortest Syntactically Correct Wizard Script

While wizard scripts are constructed by responding to prompts in the Script Wizard, the script is technically constructed with a smaller subset of the objects available to a graphical script. The Script Wizard ensures that all scripts created and saved are syntactically correct.

The shortest syntactically correct wizard script will include:

- One panel, on the root graph. This will contain an automatically generated Continue button and does not require any questions to be explicitly defined.

- One group, Disconnect, with a shortcut property of WrapUpShortcut. This enables the agent interface Disconnect button. The group represents a subgraph with a start node and a termination node (no panels), connected with default branching.

- Two start nodes (one on the root graph, and one in the Disconnect group).
- Two termination nodes (one on the root graph, and one in the Disconnect group).
- Default branching linking all objects.

The Shortest Syntactically Correct Graphical Script

To be syntactically correct, a graphical script (or any graph on any level of a graphical script) requires only a start node (created automatically), a termination node, and default branching between those objects. Of course, such a script does not contain any properties that would be processed or displayed at runtime. Only panel contents are displayed. Adding a panel, or any configurable object, includes additional requirements.

Additional Requirements for Panels

Every panel requires at least one "node" or question to be defined in order for the script to be syntactically correct. The reason for this requirement is that every panel displayed at runtime requires end user interaction to progress the flow of the script. This interaction comes in the form of the end user responding to question user interface controls in a script. At minimum, each panel has one question UI control (a button, often labeled "Continue"). If multiple nodes or questions are programmed into a panel, then there will be one question UI control per definition, in addition to a system-generated Continue button.

The panel is the only Script Author object that visible at runtime, displaying panel text, any questions (nodes) that have been defined for the panel, and associated labels for those questions.

Additional Requirements for Groups or Blocks

When an object such as a group or block exists only as a container, it must still adhere to these rules. Thus, if the container object (whether a group or a block) requires no panels or other objects within it, it must still have a termination node, connected with a Default branch from the start node.

Furthermore, every object on a graph must contain branching, initiating from the start node, in order to be processed by the script and pass a syntax check.

Indeterminate Branch Exceptions

Graphs that contain an Indeterminate branch introduce two exceptions:

1. An Indeterminate branch contains an action or expression that must be evaluated in order for the flow to be completed. A graph with an Indeterminate branch need not adhere to the requirement for a termination node in order to pass a syntax check. While a termination node is not *required*, it may be used without causing issues.

2. Indeterminate branches may contain Java code to "jump" the user to a particular panel (on the same graph) or group (anywhere in the script). Thus, when using Indeterminate branches you may use panels or groups that are not attached from above with branching. These must still be branched appropriately from that point onward on the graph.

Oracle Scripting Users

This section includes the following topics:

Script Author Users

The Script Author development environment is the component of Oracle Scripting which provides the sole means of creating scripts for execution in any Oracle Scripting runtime Scripting Engine interface.

As of Oracle Scripting release 11.5.9 or later (or Interaction Center Family Pack P or later), Script Author is a Java applet accessed through Oracle Applications by a user with the Script Administrator responsibility. In previous versions, Script Author was a standalone Windows Java application that required separate implementation, installation, and setup. Regardless, users that access Script Author are referred to as script developers.

The Script Author provides a graphical user interface intended for the functional user with some technical knowledge.

The graphical development environment provides for reuse of defined commands and existing Scripting components. The script developer will define these commands. Hooks in the Script Author UI reference technical components (for example, Oracle Forms, custom Java methods, and PL/SQL packages stored in the database) which are primarily developed externally. These components provide sophisticated functions to accomplish the functionality most enterprises want. They must be developed by individuals certified and knowledgeable in the relevant technologies. The script developer must work with these highly technical resources to ensure the custom components are appropriately integrated into the script. The script developer must also ensure the code is appropriately loaded in the database or applications server (based on approach) and properly referenced in the script. This will ensure the code is available to the base Java classes that provide Scripting runtime functionality for the Scripting Engine agent or Web interface at runtime.

As of Oracle Scripting release 11.5.9 or later (or Interaction Center Family Pack Q or later), Script Author includes a Script Wizard component accessed through the menu or Script Author toolbar from Script Author. With this new feature, less technical users can quickly and easily create simple scripts or surveys by providing script information in a series of windows known as a wizard.

Script Wizard users have limited access to the more technical features available in the graphical development environment. This provides the opportunity to divide script development amongst, for example, business process engineers with full knowledge of the business and flow requirements of a script, after which they can turn the script over to more experienced developers to add hooks, commands, reference Java or Forms, and so forth.

Users of Script Author include campaign administrators, Java developers or database programmers, business process engineers, and experienced interaction center agents with technical aptitude. The keys to successful script development are: (1) familiarity with how Oracle Scripting captures and processes data, (2) knowledge of associated technologies (including access to experts in these technologies), and (3) adequate training and familiarity with existing documentation.

Using the standalone Script Author Java application, no responsibility was required to launch Script Author on a Windows client. For current and future releases, only the Script Author Java applet, accessed through a validated Oracle Applications session, is supported.

Since script developers can deploy, delete, and otherwise affect scripts in production, and can manipulate information in the applications or other enterprise database, Oracle strongly recommends that only trusted users be provided with the Scripting Administrator responsibility.

Scripting Administrative Users

Users of the HTML-based Scripting Administration console include script developers (see Script Author Users, page 2-29), who launch the Script Author applet from the Home tab, and administer deployed scripts and custom Java archive files from the Administration tab. Interaction center campaign administrators or system administrators (as well as script developers) will also typically access this console to run panel footprint reports to help tune a script's structure, increase agent performance and reduce average talk time. To access the Scripting Administration console user interface from the CRM Home Page login (or the Single Sign-On login, if implemented), these users must have the Scripting Administrator responsibility.

Scripting Engine Users

End users of the Scripting Engine agent interface are trained interaction center agents ("agents") or customer service representatives. These are non-technical users who have received simple but thorough training in the Java-based Scripting Engine interface. These individuals include call center agents taking or presenting information over the telephone, as well as interaction center agents taking advantage of other media such as intranets, enterprise portals over the World-Wide Web, and so forth.

Scripting Engine agent interface users typically launch scripts from a business application such as Oracle TeleSales or the Customer Support component of Oracle TeleService. For testing purposes, the agent interface can also be launched in "standalone" mode. Agent interface users must have access to the appropriate Oracle Applications responsibility to launch the integrated business application from which Oracle Scripting is integrated.

End users of the Scripting Engine Web interface include users of targeted (list-based) or standard (non-list-based) survey campaign deployments, or self-service Web application users. Survey respondents may have been invited to participate in a survey through an e-mail message or through navigation to a survey-enabled site. Self-service Web application users access a script or survey in a Web browser through a self-service Web application scenario such as Oracle iSupport.

To access a survey using a self-service application, the appropriate responsibility to access that application is required. No Oracle Applications responsibilities are required of the end user to execute a script as a standard or targeted survey deployment. The user simply accesses the given survey URL in any Oracle Applications 12 certified Web browser.

Survey Administrative Users

Users of the JSP/HTML-based survey campaign administrative console are non-technical users with access to detailed project requirements. These individuals are typically interaction center survey campaign administrators or system administrators.

To access the Survey Campaign administrative interface from Oracle Personal Homepage (PHP) login (or the Single Sign-On login, if implemented), these users must have the Survey Administrator responsibility.

Planning Scripts

There is no single correct manner in which to create scripts using Script Author. However, certain information is essential to create a successful script, and a recommended flow of events is described in this section to help you use Script Author effectively.

Other tasks are recommended to assist you in properly planning for a script development project.

This section includes the following topics:

Recommended Script Creation Flow

The specific major tasks associated with creating and deploying a script are listed as follows.

1. Obtain the requirements for a script., page 2-33
2. Determine your script creation method., page 2-34
3. Create a new script.
4. Define global script attribute, such as:
    1. For wizard and for graphical scripts, define the standard script properties, such as Script Name and Script Language.
    2. For graphical scripts only, also define additional script properties, such as Footprinting, Answer Collection, and Suspendable.
    3. Global script pre- and post-actions (for graphical scripts)
    4. Script information area (for graphical scripts)
    5. Shortcut buttons (for graphical scripts)
    6. Script Disconnect button (for graphical scripts)
    7. Script Suspend button (for graphical scripts)
5. Define the scripting objects needed to implement your business rules. This includes:
    1. For graphical scripts, inserting appropriate script objects.
    2. For wizard scripts, adding panels.
6. Define properties for each configurable script object. This includes:
    1. For graphical scripts, defining panels, groups, and blocks.
    2. For wizard scripts, defining questions, answer choices, and any required validation.
7. Ensure script flow is defined appropriately. This includes:
    1. For graphical scripts, insert appropriate branching to meet flow objectives.
    2. For wizard scripts, defining the exit panel sequence for a panel, or defining the destination for specific answer choices.

8. Save the script to the filing system or database. This includes:
    1. For graphical scripts, saving the script to the file system, or publishing a script to the applications database.
    2. For wizard scripts, saving the script with a different name, if applicable.
9. Deploy the script to the Oracle Applications database:
    1. For graphical scripts, this may include checking the syntax of the script.
    2. For wizard scripts, saving the script and continuing to edit it using the wizard, saving the script and exiting the script wizard, saving the script and deploying it prior to exiting, and optionally creating a graphical copy of the wizard script.

The recommended script creation flow described in this section applies to Oracle Scripting scripts created using any method or combination of methods. Additionally, this recommended flow applies to scripts regardless of their intended Scripting Engine execution interface.

Obtaining Script Requirements Before Creating a Script

Every script is customized to meet the specific needs of an organization. Even if you are beginning your script development with a building block script or a best practice survey, it is likely that you will want to customize the script to use language, formatting, and flow appropriate to your organization's business objectives. One organization may require the use of several scripts, and it is likely that each of these may differ substantially, based on the intended purpose of the script.

For example, scripts expected to be used by agents in interaction centers often require more time and effort to integrate functions of the script with other business applications. In addition to serving the primary business purpose of a specific script, interaction center scripts may also be required to provide agents with methods to rebut customer concerns, cross-sell or up-sell products, or provide generic (or specific) customer service outside the scope of the intended narrow script objective.

Scripts intended for execution in a Web browser (as surveys or Web scripts) often require more time, effort, and planning for fine-tuning the specific questions and linking them to measurable business objectives. These scripts may also require substantial HTML customization to ensure the appearance of the script is carefully tailored to an organization's requirements, as they are seen by customers and not employees of the organization.

In general, efforts for interaction center scripts focus on agent productivity, whereas efforts for scripts for Web execution typically focus on extracting specific information, or on formatting and layout.

The key to any successful script development project is to clearly define your goals, engage in a process whereby all stakeholders sign off on defined requirements, perform

acceptance testing, and avoid scope creep.

Prior to inserting a single panel on the canvas of a graphical script, or defining a single panel using the Script Wizard, a script developer should have in her possession *explicit script guidance* in the form of detailed requirements. These requirements must be closely tailored to attain the script development project objectives.

Needed by Script Developers:

In order to develop scripts appropriately, the following, at minimum, must be identified in advance:

- The full set of business rules that must be enforced by the script
- Any text that must be communicated verbatim to the script audience (such as legal disclaimers, etc.)
- Decision points that cause branching in the flow of the script
- Data elements which must be captured during a script runtime session and used as a variable or otherwise processed later
- Data (table fields) which must be queried from or written to database tables

Script development project administrators must provide detailed requirements to script developers. *In addition, they should develop a detailed flowchart depicting the flow of the intended script.* With these items in hand, a script developer can begin to implement the business requirements of the proposed script using both Script Author and custom code.

Determining an Appropriate Script Creation Method

Script Author supports the creation of wizard scripts and graphical scripts. Both can be executed in any Scripting Engine agent interface.

Each method has certain benefits and limitations. Wizard scripts, for example, are created by responding to prompts in a series of windows known as a *wizard*. No programming knowledge is required in order to create a syntactically correct script and to include sophisticated features such as using preconfigured response validation routines for questions you create, or providing default answers to a question. Wizard scripts do not include any groups or blocks, other than a single Disconnect group which has no content (it serves to enable the Disconnect button in the agent interface at runtime). Wizard scripts are only opened or modified from the Script Wizard feature of Script Author. You can convert or "graph" a copy of a wizard script for subsequent modifications using Script Author's graphical tools. This is a one-way conversion. Any changes made to a graphical script can only be viewed in a graphical script. Using custom commands, changing certain global properties (footprinting, answer collection, or the suspendable option), or including Scripting Engine interface-specific runtime features requires the use of graphical tools.

Scripts created using graphical tools may include, at the outset, Scripting Engine interface-specific runtime features such as the script information panel and shortcut buttons. Custom commands or APIs can be associated from a graphical script. Groups and blocks can be created and configured, and the Disconnect group may be modified in a graphical script to contain other panels, groups, blocks, and so on as appropriate. Structured Query Language (SQL) select (query), insert, or update, commands can be associated with graphical scripts. Other database integration, in the form of Script Author commands referencing PL/SQL packages stored in the database, can be included in graphical scripts. You can also include embedded values in panel text to show information dynamically, or format text in various styles and using various alignment properties in a single panel using graphical tools.

Panel layout created or modified with the panel layout editor feature of a graphical script does not contain any configurable HTML tables. When including graphics, a reference to the image in panel HTML must be further customized externally to point to a corresponding file on a Web server (which must be accessible to the Scripting Engine at runtime), and reimported into the script. Any substantial changes to panel HTML can be made by exporting panel HTML, customizing the code, and reimporting into a panel from the panel layout editor.

Combining Script Development Methods

If planned appropriately, you can combine script development methods. For example, you can develop the primary flow of a script using the Script Wizard; add custom commands or configure default settings using graphical tools; and to better control the appearance of a script at runtime, you can export panel HTML, modify it using any HTML or text editor, and reimport the modified HTML for each panel as appropriate.

Custom Code

Custom code is used to obtain information needed during the execution of the script, or to enforce specific business rules. Custom code supported by Oracle Scripting includes Java commands, PL/SQL commands, SQL calls to the database, Scripting blackboard commands, forms commands, and constant commands.

Interpretation of Custom Code

Custom code is referenced in the script by associating commands to objects in the script (or to the script itself) before it is deployed to the database from Script Author. Based on the type of command, the code may be stored in the metadata of the script itself (e.g., constant commands or PL/SQL commands), in the Scripting session (blackboard commands), on the applications server (custom Java commands exposed to the application by defining the class path), or in the database (custom Java commands deployed to the database from the Scripting Administration console, and PL/SQL packages stored in the database).

Using Custom Java

Custom Java can be used with Oracle Scripting in two ways: executing commands at runtime, and replacing agent interface panels with Java beans.

Executing Script Author Commands

Scripts that reference custom Java methods associated with a Script Author command can execute each method as specified by the command at runtime.

This applies to all scripts executed in the Scripting Engine, using either the agent interface or the Web interface.

Replacing Panels with User Interface Java Beans

Scripts can also use custom Java beans to replace an entire panel in the runtime session on the agent client workstation. This provides agent users with extended functionality as customized in the Java bean. This bean is executed by Oracle JInitiator on the client.

This applies only to scripts executed in the Scripting Engine agent interface.

> **Note:** Replacing a single question in a panel with a user interface Java bean is no longer supported functionality.

Java Compilation and Oracle JInitiator Dependencies

> **Note:** This section includes information about Java Development Kit (JDK) and Java Runtime Engine (JRE) compatibility with Oracle JInitiator and with Oracle Applications. The information provided includes version numbers certified at the time of publication. Certification and compatibility information frequently changes. For the latest information, consult Oracle*MetaLink* or Oracle iSupport.

When using custom Java in support of Oracle Scripting, there are compilation dependencies based on specific circumstances. These are based primarily on the purpose of the Java archive in question (to provide Script Author commands or to replace panels with Java beans), and include the following environmental factors:

- The level of Java Runtime Environment (JRE) used on the Apache Web server for your Oracle Applications environment
- The level of Java Development Kit (JDK) used to compile your custom Java code
- The Oracle JInitiator version used on the client
- The Oracle Scripting architecture type used at the enterprise.

Custom Java for use as a Script Author command is executed by the Java Virtual Machine in the Scripting session. Using the Apache mid-tier architecture of Oracle Scripting, custom Java code is executed on the Apache Web server. Thus, this code must be compiled using a version of JDK that is compatible with the JRE used on the Apache Web server (this must be the same level or lower). At this time, appropriate JDK versions may include JDK 1.3, 1.2, or 1.1.8.

All scripts executed as surveys using the Scripting Engine Web interface use the JVM of the Apache Web server. Thus, this code must be compiled using a version of JDK that is compatible with the JRE used on the Apache Web server (this must be the same level or lower). At this time, appropriate JDK versions may include JDK 1.3, 1.2, or 1.1.8.

Custom Java that replaces a script panel with a user interface Java bean at runtime executes on the agent client. Thus, this code must be compiled using a version of JDK that is compatible with Oracle JInitiator on the agent client (this must be the same level or lower). At this time, appropriate versions may include 1.3 or 1.1.8.

Java Archive File Format Requirements

The source code for Java methods or Java beans must be compiled into executable class files and packaged into Java archives in one of two file formats, as described in the table below:

Format	Description	Example
JAR	Java Archive (JAR)	SOURCE.JAR
ZIP	WinZip archive	SOURCE.ZIP

Note: Oracle recommends using JAR file formats, although both JAR and ZIP file formats are supported. For the purposes of this document, the term "Java archive" applies to both file formats, assuming the archive contains appropriately compiled and packaged code.

Note: When the Scripting Administration console references Jar files or Jars, appropriately packaged ZIP files are included in this definition.

Java Applet Versus Standalone Application

Prior to Oracle Scripting release 11.5.9 (or Interaction Center Family Pack P), Script Author was available as in installable standalone client application supported on certain Windows platforms only. In current releases, you can access Script Author as a Java applet on any operating system platform supported by Oracle Applications. You must

be in a valid Oracle Applications session to use the Script Author Java applet.

Benefits

Benefits of running Script Author from an existing Oracle Applications session as a Java applet include:

- Ability to create, modify and deploy scripts through a firewall using HTTP and secure HTTP.
- Single login and authentication to the current production environment.

  Logging into the Scripting Administration console and subsequently launching Script Author enables the user to access the database instance, obtain reusable commands from the command library, deploy scripts, and access PL/SQL stored procedures or packages without specifying database connection and login information (providing the same instance is accessed).
- No requirement to implement (locate, download, unpackage and install) the Script Author component.
- Apps user name and password no longer required to deploy scripts.

Other Ramifications

- You must be in an authenticated Oracle Applications session in order to develop scripts.
- In order to deploy the same scripts to multiple environments (for example, test and production instances), you must log into each environment separately, in sequence.
- Since Script Author is no longer installed on a script developer's client installation, there is no longer a home directory for Script Author files. As a result:
  - Script Author standalone online help files, previously available in the Docs directory, is no longer available. For assistance using Script Author, refer to *Oracle Scripting User Guide.*
  - The CCTBUILDER.PROPERTIES file is now written to the Oracle JInitiator home directory.

    This file, created automatically the first time Script Author is used, and updated each time the application is closed, stores user settings such as Script Author window size and location.
  - The EN.CONTENTS and BUNDLE.CONTENTS files are no longer created or required.

    The default English strings for the Oracle Scripting application are stored in the

EN.CONTENTS file. Previously, when the Script Author standalone application was run for the first time after installation, the user was asked to download strings for a specific language from the database. If the user decided not to connect to the database, BUNDLE.CONTENTS was created from EN.CONTENTS.

In current releases, using the Java applet, localized strings for Script Author are downloaded from the database each time Script Author is launched. The applet uses the default language setting for the user and attempts to load the localized strings from the database for that language. If the strings for that language are not found, then the user receives a warning message, and the default strings in English are used.

Script Author File Management

As of Oracle Scripting release 11.5.9 or later, or Interaction Center Family Pack P or later, Script Author is a Java applet launched from a valid Oracle Applications session. In previous releases, Script Author was a two-tiered Windows-only client application that required separate database login and authentication at the APPS user level. With the introduction of the Java applet version of Script Author, these restrictions are eliminated.

Script Author Editing Forward and Backward Compatibility

There are currently three points at which backward compatibility is not allowed when opening Script Author scripts.

- Scripts saved using applet versions of Script Author (version 1.7.0.x) cannot be opened in Script Author applet versions 1.6.3.x and earlier.
- Scripts saved using applet versions of Script Author (version 1.6.2.x) cannot be opened in Script Author client versions 1.6.1.02 and earlier.
- Scripts saved using client versions of Script Author (version 1.6.0.x) cannot be opened in Script Author client versions 1.5.0.x and earlier.

A warning message will display for users attempting to save an older script in the new version, indicating that backward compatibility will be lost.

Note that older scripts can be imported as groups without change to the original older script file. This method is recommended when older files must retain their editability in client application versions of Script Author.

Runtime Compatibility

Oracle recommends that all scripts deployed to the database from the Script Author standalone application (Script Author versions prior to 1.6.2) be redeployed using the

Script Author Java applet, accessed from the Scripting Administration console.

Single Script Editing

The Script Author application can only open a single script at a given time. Opening a second script causes any open scripts to close, prompting the user to save any changes made to the file. This restriction applies to client application and Java applet versions of Script Author.

- To modify two scripts simultaneously, you must open two separate instances of Script Author (from separate Oracle Applications sessions, using two different compatible Web browsers). Objects or data will not be able to be conveyed using the clipboard when working in this fashion, although import and export capabilities are available.

- To deploy the same script to multiple environments (for example, to move a fully tested script from a development to a production environment), you must first save a copy of the script from the original environment to a local or shared file system. Then you must log out, log into the target environment, launch the Script Author applet, open the script, and deploy it to the environment.

Database Connection Information

The Script Author Java applet is accessed from a valid Oracle Applications session, and is automatically aware of the database location of scripts for that environment. Thus, opening Script Author seeded or stored commands in the command library, opening scripts from the database, and saving deployed scripts to the database all occur without the need to establish database connections.

To open deployed scripts from or deploy scripts to a different database instance, you must log into the Oracle Applications instance for that environment and launch the Script Author applet from the corresponding Scripting Administration console.

Connecting to the database from the Script Author client application (prior to Interaction Center Family Pack P or Oracle Scripting release 11.5.9) requires specification of the database host machine, port, and SID, plus use of the APPS level user name and password.

3

Understanding the Scripting Engine

This chapter covers the following topics:

- Introduction
- Scripting Engine Function
- Scripting Engine Agent Interface
- Scripting Engine Web Interface
- Web Interface and Agent Interface Layouts Compared
- Script End Users
- Footprinting and Answer Collection

Introduction

The Scripting Engine is the component of Oracle Scripting that executes Script Author scripts at runtime. The Scripting Engine is essentially a collection of base Java classes that process a script, any custom code associated with the script, and script end user interactions. This component includes two interfaces: the agent interface (a forms client) or the Web interface (a Web browser).

This section includes the following topics:

Scripting Engine Function

The Scripting Engine processes a script in runtime, including displaying the text, images, questions and prompts in the script, interpreting the flow based on end user responses and custom code.

Scripts developed with Script Author can be executed in one of two Scripting Engine interfaces: the agent interface, a Java window launched from an Oracle business application form, and the Web interface, a sequential interpretation of a script with each panel represented by one JSP page rendered in an Oracle Applications 12 compatible Web browser.

Scripting Engine Agent Interface

The agent interface is used by customer service or interaction center agents, typically from within Oracle TeleSales or the Customer Support module of Oracle TeleService.

> **Note:** Scripts can also be run in the agent interface in "standalone" mode (still from an active Oracle Applications session). However, this is recommended primarily for script testing.

Scripting Engine Agent Interface Components

The full agent user interface (UI) consists of several functional components. At the top of the UI, a toolbar appears, containing back and forward (left and right arrow) buttons.

The Scripting Engine agent interface contains panels (that may contain text and images, and that *must* contain at least one question per panel). These panels display in the central viewing region at runtime.

In addition, this user interface includes a progress panel. This feature displays the flow path of the current script, and answers provided for the active session. This allows an agent to have an overall view at all times of where they are in the script.

Also specific to this interface are the following:

- A programmable script information area, which can contain static or dynamic content (text or timers)
- A programmable shortcut button area that can display buttons containing shortcut commands (to navigate through the script to specified locations, or to associate any Script Author API)
- A status bar
- A programmable Disconnect button.

For suspendable scripts only, a Suspend button appears beside the Disconnect button.

Scripting Engine Agent Interface Component Details

This section includes the following topics:

Panel Display Area

The largest visible region is the panel display area, where agents see any panel text and question UI controls for the one or more questions programmed into the script.

The Scripting Engine application supports both a single-panel display and multiple-panel display:

- Using single-panel mode, only the *current* (active) panel appears on the screen at any given time.
- Under multi-panel display mode, the active panel will have focus, and all text and question UI controls appear as designed on the active panel only.

  Panels previously visited appear above, clearly unselected (the lettering in a panel without focus is smaller and gray, similar to a menu item that cannot be selected in a typical Windows-based application).

  At any given time, in multi-panel display mode, the agent can scroll up using a scroll bar located alongside the panel display area to see, and click on, previously displayed panels.

Agent interaction is required for each panel to progress to a subsequent panel. This comes in the form of interacting with the question UI control or controls displaying in the current panel. Question UI controls supported in the Scripting Engine include familiar question user interface control types that render in HTML forms: buttons, radio buttons, checkboxes and checkbox groups, dropdown lists and multi-select list boxes, text fields, text areas, and password fields.

After interacting appropriately with the one or more question UI controls on the panel, the agent must click the Continue button, except in the case where the panel UI control

type is button, where the following considerations apply:

- Panels whose control type is button may display one or more buttons, each of which, when clicked, can direct the script to a specific panel in the script.
- Where a button-type panel contains a single button, the button display value does not have to be "Continue"; frequently, but not necessarily, that button directs the script to the next panel.
- If several answer choices are defined for a button question UI control, the display value for each answer choice will display as a set of buttons, one of which can be clicked at runtime. Selecting that button will pass the value of the button to the Scripting blackboard as the selected answer, and progress you to the panel associated with the button.

  Note: If the script developer uses the button question UI control type in a panel, that panel may contain only a single question.

Progress Area

As the agent progresses through panels in the Scripting Engine agent interface, information for each panel appears, by default, in the progress area. This information includes:

- The sequence in which the panels are accessed for the current script transaction (in dynamically numbered panels, beginning from 1).
- The label assigned to each panel.
- The label assigned to each question in the panel.
- The script end user's response at runtime (with the exception of information entered into the Password question UI type, which appears as a series of lower case x's).

The agent interface defaults to the Show Answers setting. Click the Hide Answers option (to the right of the portion of the UI that says Progress) to hide the end user's responses and only display the panel sequence and panel label.

Information appearing in the progress area (panel sequence, questions answered, and responses to each question) is retained in the scripting blackboard for the length of the scripting transaction.

To the right of the progress area, a scroll bar appears (when necessary) when the available frame is filled with data.

The Scripting Engine application supports both a single-panel display and multiple-panel display. Using single-panel mode, only the *current* (active) panel appears on the screen at any given time.

When using multi-panel display mode, the active panel only has focus, with all text and question UI controls clearly active. Panels previously visited for the current script session appear above, clearly unselected (the lettering in a panel without focus is smaller and gray, similar to a menu item that cannot be selected in a typical Windows-based application).

At any given time, whether the agent interface is set up for single or multi-panel display mode, the agent can scroll up through the panel information in the progress area (if necessary) using the scroll bar, and click on any panel previously displayed during the current script session. Clicking the panel brings it into focus as the active panel. The selected panel again populates in the panel display area. In multi-panel display mode, the selected panel again gains focus. In single-panel display mode, the selected panel again populates the entire panel display area.

Changing Answers Using the Progress Area

By design, when an agent revisits a previous panel and changes the answer, if branching in the script does not change as a result of that resupplied answer, then the script will return focus to the *next unanswered question.*

For example, imagine a sales script in which panel one provides info on the item being offered, and asks if the customer wants to purchase. This panel has Yes and No radio buttons, and uses distinct branching. Providing a value of No in panel one takes the customer to a collection of panels to offer the customer additional values (panels fifteen through twenty). Providing a value of Yes in panel one uses a series of panels with default branching to collect customer information. Panel two collects the customer's name, panel three collects an address, panel four collects a phone number, and panel five collects a shipping method. If at panel five, the customer decides to have the item shipped to a work address instead of the home address he provided, then the agent can click on panel three and enter the revised address. When the agent clicks continue, the script will then display panel six.

However, if the customer changes his mind and decides not to order, then the agent can click on panel one and click No. In this scenario, the initial flow taken through the script is changed. Panels two through five disappear from the progress area, and panel fifteen is now the active panel in the panel display area.

Note, however, that if the Footprinting and Answer Collection options in the global properties for this script were enabled, that all panels visited (including panels two through five) are still collected in the footprinting information, and will be available for reporting.

Script Information Area

Optionally, information may be associated with a script in the area known as the *script information area*. The script information area can accommodate up to nine separate pieces of information (a string of text or a timer), including a label for each. These fields can contain dynamic information using Scripting APIs. Any information in the script information area is associated as a global script property.

The script information area appears at runtime above the panel display area. If a script

does not have information associated with the script information area, this area is empty at runtime.

The script information area was formerly known as the static information panel. The name has changed to reflect the ability of this feature to contain dynamic information. For example, if using timers, you can dynamically enable or disable timers based on conditions programmed into the script, or define buttons (in the shortcut button area) that users can click to enable or disable the timer. Best Practice Java methods supporting this ability (startTimer, stopTimer and resetTimer) are documented in *Oracle Scripting Developer's Guide*.

Shortcut Button Area

The shortcut button area appears at runtime in the Scripting Engine agent interface immediately above the panel display area, and below the script information area. If a script does not have buttons defined, the button bar is empty at runtime. If shortcut buttons are defined, they appear in this area.

Shortcut buttons are so named because they are most often used to "jump" the script end user from the current panel to a specified group elsewhere in the script. To successfully program this function, the script developer must associate a Java command referencing the shortcut with the shortcut button (as a global script command), and associate that shortcut name to the destination group in the script.

However, shortcut buttons are not limited to the shortcut functionality. A shortcut button can be associated to *any* Script Author command (Java, PL/SQL, Forms, etc.), and thus can be used to invoke any API to do any type of processing at runtime. Other common applications of the shortcut button include launching a Web browser window (set for a specific URL), launching a specific Oracle Form for Forms-based Oracle Applications, or enabling or disabling script timers in the script information area.

The function, display content, and tool tips for buttons are global script properties. In the Scripting Engine agent interface, the shortcut button area appears in every session of that script. Button properties are configured using the Shortcut Panel in the Script Author script properties window.

References

The following functionalities are documented in *Oracle Scripting Developer's Guide:*

- The jumpToShortcut API
- Best practice Java methods enableButton and disableButton, which provide the ability to enable or disable buttons in the shortcut button area.
- The procedure to define a shortcut button to launch an Oracle Form.
- The procedure to define a shortcut button to open a Web browser window.

Disconnect Button

A Disconnect button appears on the bottom right of the Scripting Engine window. In

Script Author, when you provide the WrapUpShortcut name to the Shortcut property of a group on the root graph, you create a destination to which the script end user is routed when the Disconnect button is clicked at runtime.

If the group contains one or more panels, the first panel in the group appears to the end user immediately after clicking the Disconnect button. You can use panels in this group to capture information from each customer at the close of the script transaction. Typically, if the group contains one or more panels, it is referred to as a Wrap-up group.

If the group contains no panels, and is properly branched to a Termination node on the root graph, then the result at runtime of clicking the Disconnect button is to immediately terminate the script, following the syntactic rules of Oracle Scripting for a complete transaction. Typically, this is referred to as a Disconnect group.

Scripts created using the Script Wizard automatically contain a Disconnect group that is properly terminated, and that contains no panels. This is the only group created by the Script Wizard tool.

For graphical scripts only, if you click the Disconnect button at runtime, and receive an Invalid Shortcut Exception message, the most likely cause is that a destination has not been provided for this shortcut. In other words, a group has not been assigned the WrapUpShortcut shortcut name. In this case, return to Script Author and add this property to a group. Ensure the group is properly terminated (both within the group, and on the root graph).

Suspend Button

For suspendable scripts, a Suspend button may appear on the bottom right of the Scripting Engine window, just to the left of the Disconnect button

At runtime, when you click the Suspend button in the agent interface, the script transaction ends. All information from the current script session in its current state is recorded in the database as a suspended script transaction. This transaction is associated with a scripting transaction ID.

When you resume that session in the Scripting Engine agent interface, all state in the script is restored:

- The last answered panel appears.
- All blackboard answers recorded in the suspended transaction are restored to the scripting blackboard (temporary, dynamic memory).
- The state of the shortcut button area is restored to its state at the time the transaction was suspended.
- All commands (Java, SQL, Forms, Blackboard, Constant) are re-executed upon resumption.

Display of the Suspend button is controlled in three different ways:

- IES : Display Suspend Button on Script Frame system profile (defaults to True).
- Suspendable script property (a global property of the Script Author script). Whether created using the Script Wizard or a graphical script, this feature defaults to True.
- The global PL/SQL variable GLOBAL.IES_DISPLAY_SUSPEND_BUTTON, used by integrating applications to control display of the Suspend button.

You should only suspend a script if using a business application that provides a method to resume the transaction. At this time, the only application that includes this function in its user interface is Oracle TeleSales (the eBusiness Center).

Toolbar

A toolbar appears at the top of the Scripting Engine window. From here you can navigate back to previously visited panels using the first tool on the toolbar, the Previous panel button. If you have backtracked in a script, you may move forward in the same path one panel at a time using the Next panel button, or skip to the last panel thus far visited using the Last panel button. You may also pop a Web browser using the Toggle Browser button (the same browser used to launch Oracle Applications appears in a separate window by default). The Help button is not implemented.

Scripting Engine Web Interface

The Web interface supports obtaining survey data, feedback or opinions from customers who respond to a survey questionnaire. The survey questionnaire is a Script Author script executed as a series of JavaServer pages in a Web browser.

This user interface includes only panel display and number of Oracle Scripting buttons in the Web browser window. There is no progress area, script information area, button bar, or Disconnect button.

Unlike the agent interface, the Web interface does not support multi-panel display mode.

A special Oracle Scripting toolbar appears above the panel display, which contains the following buttons: Back, Save for Later (this button appears only for suspendable scripts in suspendable survey campaigns), Reset to Default, and Next.

Users can use the Oracle Scripting Back button to visit previous panels, and change answers if desired.

Users can use the Reset to Default button to reset each field of the currently displayed panel to either the default value (or blank) for the field or, if the panel was previously displayed and values entered, the last value entered.

Setting up Scripts for the Web Interface

To execute a script in a Web browser, users can access a survey URL. To obtain a valid survey URL, a survey administrator must perform a number of extra setup procedures.

In summary, these are:

- Set up a survey campaign
- Associate the script with the survey campaign
- Optionally identify JSP components known as survey resources
- Define a cycle and deployment for the survey campaign
- Activate the deployment

For more details, see the *Oracle Scripting Implementation Guide*.

Running Scripts Through the Web Interface

- Survey data can be solicited by sending out e-mail invitations and reminders, leveraging Oracle Marketing's list management capabilities and Oracle One-to-One Fulfillment's e-mail template, data field merging, and delivery capabilities. These are known as targeted or list-based survey deployments.
- Survey data can also be solicited by links on an enterprise Web site.
- Finally, survey data can be solicited by customized links on self-service Oracle Application user interfaces such as Oracle iSupport.

Each of these models relies on a survey URL that defines the Oracle Applications instance, and identifies (at minimum) survey campaign and deployment identification code (dID) parameters. Other potential parameters in the URL include a respondent identification code (rID, which associates an individual with a list record), and any custom parameters to pass information into the Scripting session for evaluation and use during the interaction. Accessing this URL launches the script in the Scripting Engine Web interface.

The Scripting Engine interprets the respondent's responses to questions in the script, processes the script, and executes any custom code, passing metadata and instructions to the Web browser to be interpreted on the respondent's client computer.

Scripting Engine Web Interface Components

The components of the Web interface are as follows:

- Panel Display Area
- A toolbar containing Back, Reset to Default, Next, and optionally, Save for Later buttons

For more information about these components, see Scripting Engine Web Interface Component Details, page 3-10.

For more information about using the Back, Next, and Save for Later buttons, see the

following topics:

- Navigating the Web Interface, page 9-9
- Suspending and Resuming in the Web Interface, page 3-10

Scripting Engine Web Interface Component Details

This section describes the panel display area and the buttons that appear in the Web interface, and the general usage of each button.

Panel Display Area

In the Web interface, the panel display area shows the current panel only. Typically, you answer the questions in the panel, then click one of the Oracle Scripting buttons, such as Back or Next, to progress through the script.

Back Button

The Back button enables you to go back to the previous panel in your script flow. This button is disabled if the displayed panel is the first in the script.

Save for Later Button

The Save for Later button only appears for scripts running in suspendable survey campaigns. Clicking the button causes the script to be suspended.

Reset to Default Button

The Reset to Default button allows you to discard the answers entered so far in the panel during the current session. If this is the first time that any answers have been entered in the panel, when you click Reset to Default, each field of the panel is set to a default value or blank; for all other cases, the fields are set to the last answers entered in a previous display of the panel.

Next Button

The Next button is the standard mechanism used to progress through the script. Answers entered on the panel are sent to the Scripting Engine, which processes them, and determines the next panel to display.

Suspending and Resuming in the Web Interface

To be able to suspend and resume a survey campaign script, both the script and the survey campaign must have been set up as suspendable.

> **Note:** The facility to suspend and resume Oracle Scripting transactions in the Web interface applies mainly to targeted surveys. Oracle applications that launch standard surveys may implement the facility

to allow users to suspend and resume scripts launched from those applications.

Suspending a Survey Campaign Script

When you run a script that has been set up in a suspendable survey campaign, the Web interface includes a Save for Later button.

At any time during the Web interface scripting session, you can click the Save for Later button to suspend the script. Any changes you have made on the displayed panel at that point are saved, but not submitted to the Scripting Engine.

There are two further cases where Oracle Scripting suspends a survey campaign script:

- Timeouts

  If a survey campaign script times out, it will be automatically suspended.

- "Always Suspend" Survey Campaigns

  Generally, when you reach the end of a script, the script is marked as complete and cannot be suspended subsequently

  A survey campaign can be set up with the "Always Suspends" attribute. When you run a script in such a campaign, every termination of that script, whether you time out or reach the end of the script, results in the script being suspended.

In both of these cases, as with the manual case where you click Save for Later to suspend your script, your script answers are saved, but not submitted to the Scripting Engine.

Resuming a Survey Campaign Script

In general, Oracle Scripting has the facility to resume at one of the following panels:

- The panel of the survey where the script was suspended
- The first panel of the survey: this is to allow you to review and possibly update the answers given previously
- The last panel reached in the previous session

> **Note:** Frequently, the last panel is also the panel where the session was suspended. However, in the session where the survey was suspended, the script user could have navigated backwards from the last panel and suspended the survey with an earlier panel displayed.

The resume options available to users depend on how the Oracle application that controls script launching has been set up.

Respondents of a suspended target survey can resume the survey by clicking the same

URL that they used to launch the survey originally. Oracle Scripting will present them with a page that displays the date and time that the survey was suspended, and ask if they want to resume the survey from where it was suspended or to start the survey afresh from the beginning.

Uses for the Scripting Engine Web Interface

There are several applications of a script in this interface. In all cases, the primary purpose is to exchange information between parties.

- If the purpose is strictly to gather information or opinion, scripts can be executed as online survey questionnaires. In this case, the survey URL is provided to potential survey respondents.

- Integrating with Oracle Marketing and Oracle One-to-One Fulfillment, information can be surveyed from a *known* survey sample (a targeted audience, as represented by a list in Oracle Marketing). This is known as a targeted survey, and requires additional administration steps, as documented in *Oracle Scripting Implementation Guide*. In this case, all list members (and only list members) are potential survey respondents. If multiple lists are used, each is implemented as a separate deployment.

- When scripts are enabled from an Oracle self-service Web applications, they are typically known as Web scripts. This requires modification of the application user interface to embed a valid survey URL.

Execution of a script in the Web interface requires an active, authenticated Oracle Applications session. If the user is already logged into an Oracle Web-based self-service application such as Oracle iSupport, authentication information for the current session is used. If invoking this URL by responding to an invitation or reminder, or by clicking on a link from the polling enterprise's Web site, an Oracle Applications session is initiated over the hypertext transfer protocol (HTTP), using an applications-level guest user login.

Survey Resources

Survey resources are files that can display during execution of a script in a Web browser. Header section and footer section pages display on each panel, if included in the survey campaign requirements. The error page or final page resources display upon server-side error conditions, or after the last panel is displayed, respectively.

Scripts are executed in the Scripting Engine Web interface using one of two technology stacks: the deprecated Oracle CRM Technology Foundation (JTT) technology stack, and the new Oracle Applications Framework technology stack. The tech stack executed at runtime is determined by the base tech stack option (a survey campaign level requirement) established by the survey administrator. There are some differences in survey resources and requirements.

For scripts executed in a Web browser using the JTT tech stack, resources are JavaServer Pages (JSP) files. These campaigns require header page, error page, and final page resources. Footer page resources are the only optional resource.

For scripts executed in a Web browser using the OA Framework tech stack, header and footer pages are optional. If you do not designate one, it simply will not be included at runtime.

If you do not designate error or final page resources, a default resource will be provided by the application at runtime, when appropriate. You can also designate a URL redirect for a final page or error page resource.

In order to assign survey resources to a survey campaign, they must first be defined. Survey resources are defined in the Survey Resources tab of the Survey Administration console.

After they are defined, you can assign a survey resource to a survey campaign for execution in the Scripting Engine Web interface. The assigned resources display (as appropriate) at runtime for all active deployments in that survey campaign.

Survey resource administration and survey campaign administration are detailed in *Oracle Scripting Implementation Guide.*

Parameters Passed to URL For Redirect Survey Resources

When defining error and final page survey resources for use with OA Framework technology stack survey campaigns, survey administrators can designate the resource type as URL For Redirect. When a fully qualified URL is invoked at runtime, the Scripting Engine redirects the browser to that URL (either when an error occurs, or at the end of the script, as appropriate).

The following key/value pairs are appended to the URL when the redirect page is called:

Information	Key	Applicability
Error information	ERR	For error pages only.
Deployment ID	dID	Error and final pages.
Transaction ID	tID	Error and final pages.
Respondent ID	rID	For targeted surveys only. Error and final pages.

If the valid URL to which the URL For Redirect survey resource points is a JavaServer Page, the JSP developer can include code within the JSP page to read the parameters passed by the Scripting Engine, and incorporate that information in the display

elements of the page.

Web Interface and Agent Interface Layouts Compared

The main script objects are the panels, and both the agent and Web interfaces display the script panels, though in slightly different ways. For survey respondents using the Web interface, the script content in each HTML page equates to the panel display area for the agent interface.

The Web interface does not contain the progress area, script information area, shortcut button area, suspend button, or Disconnect button, nor the toolbar associated with the agent interface.

The Web interface displays a toolbar with the following buttons: Back, Next, Reset to Default, and, optionally, Save for Later..

The Web interface uses optional survey resources, which have no corresponding match in the agent interface.

Ramifications of User Interface Differences

While the same script can be used in either interface, information programmed for the script information area or shortcut button area will be ignored in the Web interface. Any wrapup group should be included in the flow of a script intended to be executed in the Web interface, since there is no Disconnect button to jump to that group. Any jumps to groups containing specific functionality should be accomplished with custom code, since survey respondents will not have shortcut buttons to accomplish this purpose.

Script End Users

This section includes the following topics:

Agent Users

Interaction Center agents typically have access to at least one business application from which scripts are executed. Thus, generally, agent users are required to be members of the enterprise database and may require membership in a resource group with a sales role (necessitating importation and group administration using CRM Resource Manager). Therefore, it is recommended that you perform all three user creation and administration tasks for agent users.

Self-Service Web Application Users

Self-Service Web application users typically execute a script by selecting a survey URL link added to the customized Web application. Thus, they gain access to an Oracle Applications session prior to executing the script (when logging into Oracle Applications to use the integrated self-service Web application). From an Oracle Scripting perspective, these users have no special requirements (roles, responsibilities or functions) outside of those required for the self-service Web applications.

Web Interface Guest Users

Users of the Scripting Engine Web interface who are *not* using self-service Web applications (survey respondents or Web script users) typically gain access to an Oracle Applications session by entering a survey URL into a Web browser. A valid survey URL provides access to a guest user Oracle Applications account that is restricted to execution of a script only. The guest user is seeded with Oracle Applications; assuming it has not been disabled, this functionality requires no additional setup.

Footprinting and Answer Collection

Footprinting is the recording in the Oracle Applications database of which panels in a script transaction were visited by a script end user, in sequence, and the duration of time (in milliseconds) before the next panel is requested. Footprint data is stored in tables IES_PANEL_DATA and IES_FOOTPRINTING_DATA in the Oracle Applications schema.

Answer collection is the recording in the Oracle Applications database of end user responses ("answers") to all answer controls ("questions") that are marked in the script as collectable. Answer collection data is stored in the IES_QUESTION_DATA table in the Oracle Applications schema.

Footprinting and answer collection are both global script properties. For any given script, these features are either on or off. These options can be viewed and set in Script Author by selecting File > Script Properties.

If enabled, footprinting data and answers are collected for each transaction or session of the script running in the Scripting Engine, in either interface.

In order for footprint data to be collected, either the Footprinting option or the Answer Collection option must be selected at the global script level. Otherwise, regardless of which specific answers are marked as collectable, information for each script end user will be discarded at the end of each script session.

Footprinting and Answer Collection Dependencies

- Answers are only collected for questions designated as collectable. The Collectable option is a boolean property of a Script Author question, represented by a

checkbox. *This option is selected by default for all questions defined in Script Author.* You can modify any single question so that answers provided at runtime are not collected, by clearing the Collectable option selection in the data dictionary for that question UI control.

- Answers are only collected for scripts for which the Answer Collection option is selected. To prevent answers designated as collectable from being collected for all questions in a script, you can clear the Answer Collection option at the global script level. Doing so will prevent survey summary reporting capabilities for scripts with answer collection disabled.

- Answer collection requires footprinting. Therefore, regardless of whether the Footprinting option is explicitly selected, footprinting data will be collected for any script with the answer collection option selected.

  If the Footprinting option is selected but the Answer Collection option is not, only footprinting data for each session or transaction of that script will be saved to IES_PANEL_DATA and IES_FOOTPRINTING_DATA. If neither option is selected for a specific script, no footprinting or answer collection data is saved.

- Saving footprint data whenever answer collection is enabled ensures, for each script session, a link between each response provided at runtime, and between the specific panel instance from which that response was provided. In the answer collection table (IES_QUESTION_DATA), column PANEL_DATA_ID) contains the foreign key reference to IES_PANEL_DATA (the footprinting table).

4

Understanding the Scripting Administration Console

This chapter covers the following topics:

- Introduction
- Scripting Administration Console Features
- Oracle Scripting Administration Concepts

Introduction

The Scripting Administration console is an HTML administration user interface for script developers and administrators.

This section includes the following topics:

Scripting Administration Console Features

The Scripting Administration console is an HTML administration user interface for script developers and administrators, which relies upon the Oracle CRM Technology Foundation (JTT) technology stack.

The Scripting Administration console has three primary functions: to launch the Script Author Java applet, to provide administration of Oracle Scripting files, and to provide access to agent application reports.

This section includes the following topics:

Script Author Applet

From the Home tab, logged-in users of the Scripting Administration console can launch Script Author as a Java applet. No additional login information is required to launch the applet, connect to the database, access the command library, or deploy scripts.

Oracle Scripting File Administration

From the Administration tab, administrators can administer deployed scripts and Java archive files used by Oracle Scripting. Specifically, you can perform the following:

- View and delete deployed scripts.
- View, upload, update, and remove custom Java archives in support of Scripting operations.
- Set and remove the Global property for uploaded JAR files. The global property enables a specified set of code to automatically be loaded and available to all active scripts.
- Map Java archives to specified scripts to enable that code to be explicitly loaded and available to the script.

This console is accessed by logging into Oracle HTML-based applications using a user account with the Scripting Administrator responsibility.

Oracle Scripting Agent Interface Reports

From the Reports tab, you can generate and view panel footprint reports for a specified script. This report is instrumental in tuning a script, and indicates for each session or interaction of a script which panels were visited and the duration of the visit (in milliseconds). This is currently the only agent interface report for Oracle Scripting.

To generate meaningful information, the designated script must collect footprinting and answer collection information. These are global attributes of a script. At minimum, the Answer Collection property must be selected (this will also result in the collection of footprinting data).

Footprinting and Answer Collection

Answer collection is the recording of end user responses ("answers") to all question UI controls ("questions") that are marked in the script as collectable. Answers are collected for each transaction or session of the script running in the Scripting Engine, in either interface. If enabled (at the script level), answer collection data is collected in table

IES_QUESTION_DATA.

Footprinting is the recording in the database of which panels in a script transaction were visited, and the duration of time in milliseconds before the next panel is requested. Footprint data is stored in two table, IES_PANEL_DATA and IES_FOOTPRINTING_DATA.

For each script, you can designate the collection of footprint data by enabling the Footprinting option as a global script property. For each new script created, this option is selected by default. These options are also enabled automatically for any script created using the Script Wizard.

Additionally, regardless of whether the Footprinting option is enabled, footprinting data is now also automatically saved to the database when the Answer Collection option is enabled. This change (introduced in Interaction Center FP-Q and later or release 11.5.9 or later) improves the quality of data saved from a script transaction for reporting purposes. This ensures a link between each response provided at runtime, and the specific panel instance from which that response was provided. Accordingly, an additional column (PANEL_DATA_ID) is contained in the answer collection table, IES_QUESTION_DATA. This column contains the foreign key reference to the footprinting table, IES_PANEL_DATA.

If the Footprinting option is selected but the Answer Collection option is not, only footprinting data for each session or transaction of that script will be saved to footprinting tables. Obviously, if neither option is selected for a specific script, no footprinting or answer collection data is saved.

If neither footprinting nor answer collection are enabled for a script:

- No data will be available from which to view individual responses from the Responses tab of the Survey Administration console.
- No data will display in the panel footprint report executed from the Reports tab of the Scripting Administration console.

If answer collection is disabled for a script, but footprinting is enabled:

- No data will be available from which to view individual responses from the Responses tab of the Survey Administration console.
- Footprinting data will still be recorded in the IES_PANEL_DATA and IES_FOOTPRINTING_DATA tables if enabled for the script.

Oracle Scripting Administration Concepts

Oracle Scripting provides the ability to create, modify, and deploy scripts (using the Script Author component) that can be executed in the Scripting Engine component. The Scripting Engine has two interfaces (the agent interface and the Web interface), both of which display the script at runtime for their intended audience. Each runtime interface

interprets the script, end user input, and any custom code associated with the script. If using the Web interface, you must use the Survey component to create and administer guidelines for the script to be executed within a Web browser.

Scripts are deployed to the applications database using Script Author. Scripts may rely on custom Java code, compiled and deployed as Java archive (JAR) or zipped archive (ZIP) files and referenced in the script. Scripts can also reference PL/SQL procedures stored in the applications database.

In support of Scripting operations, you can use the Scripting Administration console to access Script Author as a Java applet; administer scripts and script Java archive files; and monitor Scripting Engine agent interface reports.

This section includes the following topics:

Scripting Administration Console

The Scripting Administration console provides script administrators the interface to launch Script Author as a Java applet, to view and delete deployed scripts, to view and administer Java archive files, to map Java archive files to specific scripts, and to view agent interface reports. This console is accessed by logging into Oracle HTML-based applications using a user account with the Scripting Administrator responsibility.

Scripting Administration Console View List

When you view any page displaying a summary list in the Scripting Administration console, the set of records which displays in the list is filtered by the parameter selected in the View list. By default, only items created by the Oracle Applications user account with which you are currently logged in display in each summary list. To view items created by all users, change the value in the View list. When you select a filter option from the View list, the page refreshes. The summary view list displays, listing only the objects that meet the selected criteria.

If displaying a list of Java archive files on the Jar Listings or Jar Mapping pages, for example, the value My Jars is the default, resulting in the display of all JAR and ZIP files uploaded using the Scripting Administration console with your user name. To display a list of *all* Java archive files deployed from the Scripting Administration console (including those uploaded by other users in this environment), select All Jars from the View list.

If using the View menu to display a list of deployed scripts, additional filtering criteria is available based on a script's active or inactive status.

Note: Scripts are deployed to the IES_DEPLOYED_SCRIPTS table of the applications database from Script Author. Active scripts, which can be executed by any Scripting Engine of a compatible code level, contain a value of "1" for the ACTIVE_STATUS field within that table. Inactive scripts contain a value of "0" in this table field. Deployed scripts with ACTIVE_STATUS set to "0" are retained in IES_DEPLOYED_SCRIPTS so that existing footprinting and answer collection data can maintain valid references.

If displaying a list of deployed scripts on the Deployed Scripts page, the value My Active Scripts is the default, resulting in the display of all active scripts created with your user name.

To display a list of all scripts created with your user name regardless of active status, select My Scripts.

To display a list of all active scripts created with any user name, select All Active Scripts.

To display a list of all scripts created with any user name, regardless of active status, select All Scripts.

Note that if no objects meeting the filter parameter for a certain category are found, then the list headings will appear for the summary table, with no records listed. As soon as an object is created meeting that criteria, it will appear in a refreshed list.

Agent Interface Reports

The ability to report and analyze scripts executed in the Scripting Engine agent interface is critical; enterprises can use information collected in the applications database for various purposes. Reports from information collected in scripting-specific tables of the applications database as a result of Scripting operations, and other data collected from customized scripts and stored in custom tables, can be generated using any analytical tool such as Oracle Discoverer or Crystal Reports. Additionally, the Scripting Administration console provides access to panel footprint reports compiling footprinting data.

Additional reporting for scripts executed in the Web interface is available as part of Oracle Interaction Center Intelligence reporting functionality. Implementation of these reports, and access to the Oracle Discoverer tool, is required. These additional reports require the use of Scripting-specific concurrent programs and summary tables specific to survey operations. For additional information, see Oracle Scripting Survey Concepts.

This section includes the following topics:

- Required Report Parameters, page 4-7

Reports and Data

At this time, the only report available through the Scripting Administration console is a panel footprinting summary report. There are two requirements for receiving and reporting data in the Scripting Administration console:

1. In order to appropriately view reports, two script-level parameters should be enabled. These parameters, Footprinting and Data Collection, are established in the global script properties from Script Author prior to deploying a script.

   Note: Technically, footprinting is enabled when the data collection option is selected. However, Oracle recommends explicitly enabling the Footprinting option if footprinting information is desired for the purposes of reporting.

2. To generate a meaningful report, data to be displayed in the report must already be generated. Therefore, scripts must be executed in order to tabulate data displayed in the report. Each time a script is executed (assuming the appropriate parameters are enabled), data regarding the paths taken in the script (footprinting) and the answers selected during the script session (answer collection) are collected in IES tables in the Oracle Applications database.

   For a panel footprint report, this signifies a script with footprinting specified has been executed at least once, either in the interaction center interface by an agent running through a script, or in the Web interface as a respondent participates in a survey using a Web browser.

Analysis and Tuning with the Panel Footprint Summary Report

The panel footprint report may be used either for analysis of surveys, or of scripts in use in the interaction center.

Cost Savings in the Interaction Center

This report is overtly useful to enterprises using scripts in the interaction center, as footprint analysis can lead directly to reducing average talk time for an interaction center agent. Doing so results in measurable reduction in costs and increased agent efficiency.

Disabling Footprinting

Footprinting (the act of recording which panels in a script were visited and for how long) can provide useful script tuning data. However, it also consumes system resources. If an enterprise does not need to view individual responses or generate reports, then footprinting should be disabled at the script level to conserve system resources. Note that even if footprinting is not specifically enabled, but data collection is

enabled at the script level, then footprinting data will be collected in order to maintain the integrity of data collected.

Required Report Parameters

The reports take the respective parameters listed below.

Report Type	Parameters Required to Run Report
Panel Footprint Summary Report	Script Name
	Start Date
	End Date

5

Understanding Survey Campaigns

This chapter covers the following topics:

- Survey Campaign Administration Concepts
- Survey Campaigns Supported in Two Technology Stacks
- Survey Campaigns
- Statuses for Survey Campaigns and Deployments
- Survey Resources
- Survey Campaign Setup Examples
- Survey Administration Console
- The Survey URL
- Survey Reports
- Prototypes
- Concurrent Programs Supporting Survey Operations
- SUBMIT GROUP FM REQUEST FROM IES Concurrent Program
- Summarize Survey Data Concurrent Program
- Update Deployment Status Concurrent Program

Survey Campaign Administration Concepts

Using the survey component of Oracle Scripting, enterprises can

- Create, manage, and report on surveys to evaluate customer satisfaction
- Receive customer input on new initiatives
- Gain feedback from survey respondents

The data returned from surveys can then be analyzed and used to improve product

lines, target new or improved services, or otherwise improve responsiveness.

The survey component of Oracle Scripting uses Script Author scripts as Web-based survey questionnaires that can be executed in a Web browser (over the Internet or on an intranet).

> **Note:** The survey component of Oracle Scripting was previously known as iSurvey.

A survey respondent participates in a survey questionnaire by accessing a specific survey deployment URL either from an enterprise's Web site, from a self-service Oracle Application such as Oracle iSupport, or from an invitation e-mail message inviting survey participation.

The collection of requirements for conducting such an effort is referred to as a survey campaign. Survey campaigns are set up and managed from the Survey Administration console.

This section consists of the following topics:

References

- For information on performing survey administration tasks, see the Survey

Resources Administration Tasks and Survey Campaign Administration Tasks sections of *Oracle Scripting Implementation Guide.*

- For specifics on marketing list administration or fulfillment processing, refer to product documentation for Oracle Marketing and Oracle One-to-One Fulfillment, respectively.

Survey Campaigns Supported in Two Technology Stacks

Oracle Scripting supports two technology stacks for executing scripts in a Web browser:

- Oracle Applications (OA) Framework
- Deprecated - JTT

When a survey campaign is established using the OA Framework technology stack, the questionnaire script, when executed in the Scripting Engine Web interface as a survey or a Web script, runs using the OA Framework technology stack.

If the technology stack for the survey campaign is Deprecated - JTT, the questionnaire script at runtime executes in a Web browser using the Oracle CRM Technology Foundation (JTT) technology stack.

> **Note:** The technology stack assigned to a survey campaign cannot be changed after the survey definition is saved.

Oracle strongly recommends that all Oracle Scripting customers use the OA Framework functionality. The older JTT technology stack is deprecated and will be obsoleted in a future release.

Survey campaigns using the OA Framework technology stack have more options for administering survey campaign resources than for survey campaigns using the deprecated JTT technology stack, but the customer experience for executing scripts is essentially identical at runtime.

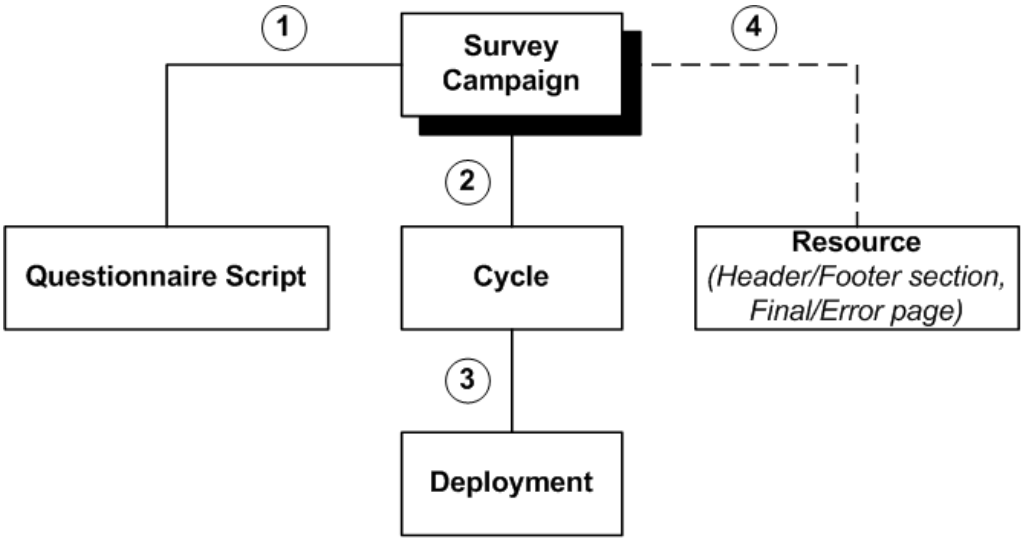
Survey Campaigns

The Survey component relies on the concept of a survey campaign: a focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose. For example, an enterprise could polling a target market segment or population, analyze the data, and then act on the results.

Typical actions might include the offering of a new product or service, or a change in business processes to improve satisfaction. As part of a self-service application, a survey campaign typically measures satisfaction regarding the customer experience for that self-service application.

Survey campaign goals are achieved by two processes:

- Creating a survey questionnaire - a script built with the campaign goals in mind
- Administering the survey campaign



1. **Each survey campaign must have one questionnaire script**
   (the same script can be associated with many survey campaigns)
2. **Each survey campaign must have one or more cycles**
3. **Each survey campaign cycle must have one or more deployments**
4. **Each survey campaign can have resources**
   (the same resource can be associated with many survey campaigns)

The top-level data structure, called a survey campaign, identifies a survey questionnaire script and global information such as which survey resources will be used at runtime.

Each survey campaign must have a questionnaire script. For more details, see Survey Questionnaires, page 5-5.

A survey resource is an object that displays during script execution, either as part of a page (header or footer section) or as a special page (error page or final page). Depending on the technology stack, the survey campaign may be required to have certain survey resources.

- If executing surveys campaigns using the Deprecated JTT technology stack, header section, final page, and error page survey resources are required; footer sections are optional.
- If executing surveys campaigns using the OA Framework technology stack, you can explicitly associate survey resources to the survey campaign, but you do not have to.

For more details, see Survey Resources, page 5-13.

The survey campaign has two lower level data structures or child objects: cycles and deployments:

- Each survey campaign must have at least one cycle, and can have more.
- The object that is actually executed is the deployment. For a survey to be run, you must create a deployment for a cycle, and set the deployment status to Active. You can create several deployments for each cycle.

Deployments can be grouped into cycles to execute the same questionnaire and parameters over a different timeline for comparison purposes.

Child objects can be added to existing parent objects. For example, for an active or open survey campaign, you may add a second cycle, or additional deployments.

> **Note:** Cancelling a cycle or a deployment does not affect the parent objects.

For more details, see the following topics:

- Cycles, page 5-6
- Deployments, page 5-6
- Survey Campaign Setup Examples, page 5-22
- The Survey URL, page 5-24

See Also

- Survey Campaign Statuses, page 5-8

Survey Questionnaires

The survey questionnaire is a Script Author script that is associated with the survey campaign. The script must be completed, tested, and deployed to the applications database to make it available for a specific survey campaign.

The individuals participating in a survey ("survey respondents") can be:

- Customers or prospects viewing the survey questionnaire using any Oracle Applications certified Web browser
- Individuals executing a Web script from an Oracle self-service application such as Oracle iSupport
- Internet users who access a valid survey URL or a link to a valid survey URL

- Members of an Oracle Marketing list who have received an invitation or reminder by e-mail to execute a survey in their browser.

On the end user's desktop, the user is guided through the script either through a prescribed order, or through a dynamically determined path - depending on the needs of the enterprise.

On the Apache Web server associated with the Oracle Applications instance for the enterprise conducting the survey or hosting the Web script, all business logic is executed dynamically. This includes the business rules embedded in the survey questionnaire script (such as rules-based branching, data integration, and commands associated with specific objects and events), the end user's answers, and any custom Java or PL/SQL code.

Cycles

Each survey campaign must have at least one cycle. It may have as many as required.

A cycle is a small collection of requirements for grouping deployments for the purpose of result reporting and analysis. Should an enterprise have a need to compare survey data for the same set of questions for a similar group of respondents (the sample) over different time periods, it can execute two separate cycles, and then compare reporting information from each successful deployment by comparing the cycles.

Cycles are defined in the Survey Administration console for any open, active, or idle survey campaign.

Deployments

When you run a survey, you actually execute a survey deployment. Each survey cycle must have at least one deployment. It may have as many as required.

For all deployments, you must provide information about execution dates for the survey campaign, and, depending on deployment type, you may be required to specify other details.

There are two types of deployment:

- *Standard - or non-list-based*

  These deployments are anonymous, and are executed by publishing, announcing, or embedding the resulting survey URL in e-mail messages or links.

- *Targeted - or list-based*

  These deployments include many additional details, including the target population (as represented by an Oracle Marketing list), and a method of delivering the resulting survey URL (specifically, Oracle One-to-One Fulfillment master documents to invite and remind list members to participate in the survey campaign)

For more details about deployments, see the following topics:

- Common Deployment Components, page 5-7
- Targeted Deployment Components, page 5-7

See Also

- Deployment Statuses, page 5-10

Common Deployment Components

Each deployment requires:

- A parent survey campaign and cycle
- A deployment name
- A media type (currently Web)
- A status
- A deployment start date and time (equal to SYSDATE or in the future)
- A response end date and time (in the future)
- A deployment type (standard or targeted)

Targeted Deployment Components

For list-based deployments only, you must provide:

- List name
- Maximum number of responses per person
- Target response percentage
- Hosting option (standalone or menu-based)
- Survey Web URL (automatically generated from logged-in Apache Web server)
- Invitation template name (identifying invitation master document)
- Invitation e-mail message subject heading

If using reminders in addition to invitations, you must also provide:

- Reminder template name (reminder master document)

- Reminder e-mail message subject heading
- Number of reminders
- Reminder interval (in days)

Statuses for Survey Campaigns and Deployments

This section consists of the following topics:

- Survey Campaign Statuses, page 5-8
- Deployment Statuses, page 5-10

Survey Campaign Statuses

Status	Description
Open	When it is initially created, a survey campaign is open and remains in this state until the first deployment is activated.
Active	When any deployment is activated, the parent survey campaign status changes from Open to Active. Once Active, a survey campaign status may change to Idle only. Its status never returns to Open, nor can it be set to Closed prior to changing to Idle. Active is only valid from the Open or Idle status. A survey campaign status cannot be manually set to Active by the survey administrator; this status is established by the system based on these business rules.
Idle	A survey campaign status is Idle if at least one of its deployments was activated in the past (the survey campaign status was previously Active and the deployment status was previously either Pending or Active), but the survey campaign currently has no active or pending deployments. Idle is only valid from the Active status. A survey campaign status cannot be manually set to Idle by the survey administrator; this status is established by the system based on these business rules.

Status	Description
Cancelled	If a survey campaign status is Open (after it is created), but has either had no deployments defined or has had no deployments activated, the survey campaign status can be set to Cancelled, indicating that its purpose is no longer relevant. Once cancelled, no updates can be made to the survey campaign or any of its children objects (cycles or deployments). Cancelled is only valid from the Open status. Only a survey administrator can change the status of an open survey campaign to Cancelled.
Closed	If the status of a survey campaign is Idle (if it currently has no active deployments), and its deployments have been successfully run for the intended duration, the survey campaign status can be set to Closed, indicating that its purpose was fulfilled. Once closed, no updates can be made to the survey campaign or any of its children objects (cycles or deployments). Closed is only valid from the Idle status. Only a survey administrator can change the status of an idle survey campaign to Closed.

When a survey campaign is created, and prior to defining a deployment, the status of the survey campaign is Open. You can view the status from the Survey Campaigns page. For open survey campaigns, you can change the status from the Status list by selecting Cancelled from the list. This invalidates the survey campaign definition, ensuring it can no longer be used.

Unless you explicitly change the status of a survey campaign after creation, it remains Open until a child deployment is activated. Once a deployment is activated, the parent survey campaign status changes to Active. A survey campaign remains active as long as at least one of its deployments has the status Active or Pending. Once the survey campaign is active, the survey campaign status can become Idle, and can change from Idle to Cancelled or Closed. Cancelled and Closed statuses only result from manual intervention by the survey administrator. The status of an active survey campaign can automatically change to Idle based on events related to its deployments. For more information, see Deployment Statuses, page 5-10.

Once all the deployments for a survey campaign have executed successfully, and the survey campaign status is Idle, you can close the survey campaign from the Status list by selecting Closed from the list. Once a survey campaign is closed, no properties of the closed survey campaign, its cycles, or deployments, can be changed. Return data remains available for viewing responses, or from Oracle Business Intelligence, and survey reports can be executed for analysis of survey returned data.

Cycles are defined from the Survey Campaigns tab of the Survey Administration console for any open, active, or idle survey campaign. As with all information entered

into the Survey Administration console, the cycle name should follow any existing predefined survey campaign requirements provided to the survey administrator.

Deployment Statuses

Status	Description
Open	The deployment status is Open when the deployment is initially created and remains in this state until activated by the survey administrator.
Pending	When a targeted deployment is activated, if the deployment date and time are in the future, its status changes from Open to Pending. The deployment remains in this state until the deployment start date and time equal SYSDATE. At this time, the SUBMIT GROUP FM REQUEST FROM IES concurrent program executes, causing the fulfillment request to be submitted to the fulfillment server. Upon a successful submission of the fulfillment request, the deployment status changes to Active. Pending is only valid from the Open status, and applies only to targeted deployments
Error	Error status indicates that the concurrent program generated an error while attempting to submit the fulfillment request to the fulfillment server. This status does not include errors that occur after the fulfillment request is successfully sent to the server. If problems occur that prevent invitation or reminder e-mail messages from being sent, the deployment status remains Active, and fulfillment debugging should commence by a fulfillment administrator from the Oracle One-to-One Fulfillment administration console. Error is only valid from the pending status, and applies only to targeted deployments.
Active	The status of a standard deployment is Active immediately after it is activated by the survey administrator. The status of a targeted deployment is Active when the deployment date and time equal SYSDATE, and the fulfillment request is successfully sent to the fulfillment server by the concurrent program. Note that Active status for targeted deployments does not indicate successful delivery of invitation or reminder e-mail messages.

Status	Description
Cancelled	If the deployment status is Open, it can be manually set to Cancelled, indicating there is no more need to execute this set of requirements. After being cancelled, a deployment cannot be executed. Cancelled is only valid from the Open deployment status.
Incomplete	If the deployment status is Active, but there is a no interest on the part of the surveying enterprise to view the results, the deployment status can be set to Incomplete.
Closed	If the deployment status is Active, and the deployment has run for its intended duration or has resulted in sufficient responses, the deployment status can be set to Closed, indicating that the survey campaign purpose was fulfilled. You can also change the status of deployments from Pending or Error to Closed.

Once a deployment is created, and before it is activated, its status is Open. You can view the deployment status from the Survey Campaign Details page or from the Deployment Details page. From either of these pages you can also change the status of an open deployment by selecting Cancelled from the status list. This invalidates the deployment definition, ensuring it can no longer be used.

The only way to change the status of a standard deployment to Active is to activate it from the Deployment Details page. The Deployment Details page will refresh, with the status set to Active and a valid survey hyperlink displayed. This is true whether the deploy date is in the past or the future.

The only way to change the status of a targeted deployment to Pending is to activate it from the Deployment Details page. If the deployment date and time are in the future, the status becomes Pending, and it remains in that state until the current date, SYSDATE, occurs. If the deployment date and time are in the past, the concurrent program will attempt to submit the fulfillment request to the fulfillment server immediately. Upon successful submission of the fulfillment request to the server, the deployment status changes to Active. For targeted deployments, the deployment details does not display a survey URL, since respondent identification numbers for each list member are required.

Once a deployment is activated, and before the deployment end date is reached, its status can change from Active to Closed or Incomplete. Do this from the deployment details page.

- Use Incomplete to indicate that there is no interest in executing this deployment and viewing its results.

- Use Closed when the deployment goals have been achieved. For example, after the deployment start and end date have passed and the deployment was successfully

executed, change Active status to Closed. As another example, close the deployment if you determine that a sufficient number of responses for a deployment have been received to perform the required analysis or make the appropriate business decisions for which the survey campaign is intended.

For targeted deployments, after the deployment is activated, it may contain either a status of Pending (indicating that it is activated and the concurrent request is in the queue or its start date is in the future), Active (indicating that the concurrent request completed successfully), or Error (indicating that the concurrent program generated an error while creating and sending the fulfillment request). Deployments with a status of Error can be modified so that the error can be corrected, and when reactivated will again display a Pending status. Targeted deployments can also contain a status of Closed or Incomplete, following the same guidelines described earlier for standard deployments.

Error status cannot be set manually. This status, for targeted deployments only, indicates that the concurrent program generated an error while attempting to submit the fulfillment request to the fulfillment server.

Status Considerations for Targeted Deployments

For targeted survey deployments, once a survey deployment is activated, you must wait until the fulfillment engine completes its activity (completes the fulfillment request and succeeds in sending the invitation master documents to the outgoing mail server specified in Oracle One-to-One Fulfillment) before you will have respondents. The concurrent request ID is listed on the Deployment Details page when displayed in the Deployment view, and can be used to track the concurrent request and its status.

Survey administrators who have also been assigned the JTF role JTF_FM_ADMIN are able to view the status and history of fulfillment requests from the Invitations tab of the Survey Administration console.

> **Note:** Granting JTF roles requires the grantor to be assigned the JTF system administrator role, JTF_SYSTEM_ADMIN_ROLE, in addition to whichever other JTF role you want to assign. For this purpose, if required, you can use the seeded SYSADMIN Oracle Applications user account. For more information, see the section Understanding Users Required for Implementation in *Oracle Scripting Implementation Guide*.

When viewing status, a request status of Submitted indicates that the list was successfully sent to the fulfillment engine.

An outcome code of Success indicates that the fulfillment server was successful in sending the invitation (or reminder) master document.

For any other outcome code (e.g., Partially Successful or Failure), you will need to log into the Fulfillment Administration console to resolve. In this scenario, list members must receive and respond to an e-mailed invitation before you can expect activity for

this deployment.

To access the Fulfillment Administration console, a user must be assigned the JTF_FM_ADMIN role and the One-to-One Fulfillment Administrator responsibility. Consult with an Oracle One-to-One Fulfillment administrator if required.

Effects of Setting Deployment to Closed

Immediately upon changing a deployment's status from Active to Closed, the status for its parent survey campaign changes from Active back to Idle, *if there are no other active or pending deployments.* The survey campaign status will remain Active if other deployments have been activated and have not encountered an error.

With survey campaigns for which all deployments have been closed, you can close the survey campaign, or create another cycle or deployment if you wish to receive more responses for this survey campaign.

Closing a survey campaign or deployment signifies that the requirements or objectives for that object have been met. No new information can be received from a closed survey campaign or deployment.

From the Survey Administration console, responses can still be viewed for closed deployments.

Survey reports can still be viewed for closed deployments.

Effects of Setting Deployment to Incomplete

Once you designate a deployment as Incomplete, if there are no other active or pending deployments, the parent survey campaign status changes to Idle, and can be set to Closed unless you wish to create new cycles or deployments to obtain additional responses.

The Incomplete deployment status indicates that the requirements or objectives for the deployment are no longer relevant. No new information can be received from a cancelled deployment. While responses and reports can be viewed for this deployment, the information generated by reports is likely to be incomplete.

Survey Resources

Survey resources are objects defined as part of a survey campaign that display for scripts at runtime to users of the Scripting Engine Web interface.

Each survey campaign can include up to four survey resources. Each survey resource has a display type, that controls how the resource is used during script execution.

The survey resources, and their corresponding display types, are as follows:

Survey Resource	Display Type
Header section	Section
Footer section	Section
Error page	Page
Final page	Page

At runtime:

- Section resources appear as a section of each HTML page that represents a panel in the script
- Page resources display as separate pages, and do not represent script panels.

If assigned to a survey campaign, header section and footer section survey resources appear at the top and bottom of each page of the script, respectively.

An error page resource displays upon any server-side error condition.

After the Scripting Engine processes the last panel in the flow of a script, a final page resource displays, or the user is redirected to a URL designated as the final page.

Error and final pages do not display header or footer section resources.

This section consists of the following topics:

Survey Resource Types and Technology Stacks

The definition and use of survey resources depends on the technology stack.

> **Note:** The survey resources you define for one technology stack cannot be used for the other.

Survey Resource Types

There are three survey resource types:

- Deprecated - JSP
- HTML File Upload
- URL For Redirect

The survey resource type and the technology stacks in which they can be used appear in the following table:

> **Note:** In the survey resource topic descriptions, the standard and alternative names for each survey resource type are used interchangeably.

Survey Resource Type	Alternative Survey Resource Type Name	Technology Stack Supported	Section Display Type Supported?	Page Display Type Supported?
Deprecated - JSP	JavaServer Page (JSP)	Deprecated - JTT	Yes	Yes
HTML File Upload	HTML file	OA Framework	Yes	Yes
URL For Redirect	URL	OA Framework	No	Yes

Survey Resources and OA Framework

The OA Framework technology stack supports two types of survey resource:

- HTML File Upload
  - The survey resources can be HTML files only.
- URL For Redirect
  - The survey resources can be either HTML or JSP pages.
  - The URL redirects users to a specific Web page upon error or at the end of the script, instead of specifying specific error or final page survey resources.

Using the OA Framework technology stack, *youdo not have to define survey resources*:

- If you choose not to specify page survey resources, a default error page and a final page are provided by the application at runtime, when appropriate.
- If you do not designate a header or a footer section, each panel rendered in the Web browser will simply not include the corresponding section (or sections) at runtime.

Survey Resources and JTT

When using the Deprecated - JTT technology stack, all survey resources are JSP - HTML or redirect survey resources are *not* supported.

You must define the following survey resources for each survey campaign set up in the Deprecated - JTT technology stack:

- Header section
- Error page
- Final page

The footer section is optional.

See Also

- OA Framework Survey Resources, page 5-17
- JTT Survey Resources, page 5-18

Survey Resource Administration

You define survey resources, and assign them to survey campaigns, in the Survey Administration console.

Survey resources consist of two components:

- The survey resource definition
- The physical survey resource file or the URL where the physical survey resource is stored.

This section consists of the following topics:

- Survey Resource Definitions, page 5-17
- Physical Files and URLs, page 5-17
- OA Framework Survey Resources, page 5-17
- JTT Survey Resources, page 5-18
- Usage of Survey Resources in Survey Campaigns, page 5-19

Survey Resource Definitions

When you define a survey resource, you create a database object that points either to a physical file (HTML or JavaServer Page, based on technology stack), or a URL (for error page or final page survey resources, supported for the OA Framework technology stack only).

Physical Files and URLs

The second component of a survey resource is the physical file itself. It corresponds to the logical definition of the survey resource, and must be available at runtime.

Physical survey resource files must be created by certified, knowledgeable HTML or JavaServer Page developers.

> **Note:** *The creation of survey resources is outside the scope of Oracle Applications.*

The location of the physical survey resource file at script execution time depends on the survey resource type and the technology stack:

- Physical HTML files corresponding to a section or page survey resource definition - for the OA Framework technology stack only - are stored in the applications database. These are automatically uploaded when defining an HTML survey resource.

- JSP files corresponding to a section or page survey resource definition - for the Deprecated - JTT technology stack only - must be uploaded to the $OA_HTML directory on the applications server. You must do this manually, as you cannot do this in Oracle Scripting.

- URL survey resources corresponding to a page survey resource definition - for the OA Framework technology stack only - must refer to an absolute URL accessible to the Apache Web server at runtime. Assuming the page to which users are being redirected exists, there is no physical file required to be uploaded.

OA Framework Survey Resources

For survey campaigns defined in the OA Framework technology stack, survey resources are either URLs for redirect or hypertext markup language (HTML) files. Both .HTM and .HTML file types are supported (regardless of case). JSP survey resources are not supported for this technology stack.

For OA Framework survey campaigns, error page or final page survey resources can also be URLs accessible to the Web server at script runtime, which redirect the script end user to the specified URL during an error or upon completing the script, respectively.

OA Framework survey campaigns do not require survey resources to be referenced at runtime, although any survey resources that are referenced must first be defined. For more information, see Survey Resources and OA Framework , page 5-15and Using Default OA Framework Survey Resources for Implementation Testing, page 5-20.

A typical implementation sequence would be:

1. You define the survey resource; during the survey resource definition, the physical HTML file corresponding to the OA Framework survey resource is uploaded to the database.

2. You create survey campaigns that reference the survey resources.

3. You can change survey resources used by a survey campaign at any time until the survey campaign is canceled or completed.

> **Note:** If defining a static HTML page as an error page for OA Framework survey campaigns, you can also require error information to display on top of the error page by selecting the Display Error on Top box.

Creating OA Framework Survey Resources

Creating and modifying the physical HTML files that serve as survey resources for OA Framework survey campaigns are not accomplished from within the Survey Administration console.

Creation of the physical files for use as survey resources is outside of the scope of Oracle Applications. You can, however, upload HTML files intended for use as survey resources from the Survey Resources tab, at the time of survey resource definition. HTML survey resources are stored in the applications database and available to any survey campaign in the same environment that references the survey resource at the survey campaign level.

JTT Survey Resources

The physical survey resource files for JTT survey campaigns must be saved in JavaServer Page (JSP) format (with a file extension of .JSP), even if they contain no dynamic content.

Header section, error page, and final page survey resources must all be referenced upon creating a JTT survey campaign. Footer section survey resources are optional.

Survey resources used for JTT survey campaigns can be defined (including the file name of the physical survey resource) and referenced in a JTT survey campaign before the physical file exists, and before it is uploaded to the applications tier. However, the physical JSP files referenced by their corresponding survey resource definitions must be uploaded to the $OA_HTML directory on the applications server prior to execution of the script referenced in a survey campaign at runtime.

In support of JTT survey campaigns, four test JSP files are seeded in the appropriate directory on the APPL_TOP, and may be used to test survey functionality. Although these test survey resources exist in the $OA_HTML directory, as with all survey resources, they must still be defined prior to use.

No seeded footer survey resource sample is included with Oracle Applications. For test purposes, you can reference a header survey resource in the Footer Section field, to ensure it displays as appropriate at runtime.

For more information, see Seeded JSP Survey Resources, page 5-20.

Creating JTT Survey Resources

Creating, modifying, and uploading the physical JSP files that serve as survey resources for JTT survey campaigns are tasks outside of the scope of Oracle Applications.

Physical survey resource files must be created by certified, knowledgeable JavaServer Page developers, and uploaded to $OA_HTML on the applications tier of the applications server by a system administrator or other individual with the appropriate privileges and access to the APPL_TOP.

However, as described in the section Test Survey Resources, page 5-20, four test JSP survey resource files ship with Oracle Applications, seeded in the appropriate directory ($OA_HTML) on the applications server. You can use the four test survey resources to test your implementation of JTT survey campaigns, and to serve as building blocks, modifying copies of these files for your own use as appropriate.

> **Note:** JSP files *must* be tested on an existing Web server to view appropriately.

> **Note:** Like all customizations, any modifications you make to survey resources are not supported by Oracle.

> **Note:** Oracle recommends using individuals certified in Java development and JavaServer Pages technology to create or modify JSP survey resources.

Usage of Survey Resources in Survey Campaigns

Once defined, survey resource definitions persist; the same survey resources can be used by any number of survey campaigns (of the same technology stack) without limit.

Any survey resource that you want to reference when creating a survey campaign *must* be defined *before you create the campaign.*

You can change any survey resource associated with an open or active survey campaign after survey campaign creation.

General Survey Resource Considerations

This section consists of the following topics:

- Including Graphics in Survey Resources, page 5-20
- Using Default OA Framework Survey Resources for Implementation Testing, page 5-20

Including Graphics in Survey Resources

Survey resources, like any other HTML or JSP page, may include images and hyperlinks. Survey resources must be located in the $OA_HTML directory on the applications server to use at runtime. Any objects (such as GIF or JPG images) referenced in a survey resource page must also be available to the application server at runtime.

Using Default OA Framework Survey Resources for Implementation Testing

If you do not associate custom survey resources for a survey campaign using the OA Framework base technology stack, no header or footer sections will appear for HTML pages representing panels. In addition, a default error page and final page will appear at the appropriate time (upon error, or after visiting the last panel in a script). Omitting the section survey resources, and utilizing the default page survey resources, provides a method to verify that survey resources are appropriately displayed upon execution of a survey campaign deployment using the OA Framework technology stack.

At the same time, you are provided with a layer of abstraction in regard to the need to ensure the product is performing as expected without needing to test the creation of tailored HTML pages. Subsequent to successful execution of a script in a Web browser with all appropriate panel content and default survey resources displaying, you can assign custom survey resources to an OA Framework survey campaign and retest.

Seeded JSP Survey Resources

Survey resources must be defined in the Survey Administration console and must map to existing JSP documents residing on the server (as described in the Uploading JSP Survey Resources section of *Oracle Scripting Implementation Guide*).

For survey campaigns in production, you will probably want to create your own survey resources. For testing and implementation verification, you may want to use the survey resources seeded with Oracle Applications.

Test Survey Resources

Test JSP survey resources, identified in the table below, ship with Oracle Applications beginning with IES MiniPack OracleG, and available with any Rapid Install from

release 11.5.5 and later. These survey resources are physically located in $OA_HTML on the applications server.

Survey Resource	File Name
Header section	IESSVYTESTHEADER.JSP
Error page	IESSVYTESTERROR.JSP
Final page	IESSVYTESTTHANKU.JSP
Hosted survey error page	IESSVYMENUBASEDTESTERROR.JSP

Note that there is no seeded footer section survey resource. For testing purposes, to ensure a footer will appear as expected in a production environment, you can use another JSP page (for example, the seeded header section).

This section consists of the following topics:

- Using Test JSP Survey Resources for Implementation Testing, page 5-21
- Additional JSP Error Page Survey Resource for Hosted Surveys, page 5-21
- Sample Code for Test JSP Header Section Survey Resource, page 5-22

Using Test JSP Survey Resources for Implementation Testing

Utilizing these test survey resources provides a method to verify that survey resources are appropriately displayed upon execution of a survey campaign deployment using the JTT technology stack in an HTML user interface. At the same time, you are provided with a layer of abstraction in regard to the need to ensure the product is performing as expected without needing to test the creation of tailored JSP pages. For this purpose, for JTT survey campaigns it is recommended that you first test execution of scripts in a Web browser using seeded test survey resources listed earlier, to ensure successful implementation. Subsequent to successful execution of a script in a Web browser with all appropriate test survey resources displaying, you can change the survey resources assigned to a survey campaign to use customized survey resources.

Additional JSP Error Page Survey Resource for Hosted Surveys

Header section and final page seeded JSP survey resources are intended for use with standalone and menu-based survey deployments for JTT survey campaigns. The IESSVYTESTERROR.JSP error page is specifically intended for use with standalone deployments. For hosted scripting operations (self-service Web applications such as Oracle iSupport, or other self-service Web applications customized to provide access to scripts executed in a Web browser), the IESSVYTESTERROR.JSP error page should be

used.

To test a hosted scenario, you can use any JSP header section (or HTML page saved in JSP file format). Again, you can use the seeded test header section to test footer section functionality as well. Error and final pages must also be in JSP file format. The error page must adhere to the hosted template. There are two ways to handle the final page:

1. Define as the final survey resource page the JSP page which displays the list of URLs to take the survey. In this way, when the survey is completed, the respondent will be returned to the starting page.

2. Define another JSP page as the final page survey resource, but include in that page a JSP forward command to point the user back to the page that displays the list of URLs to take the survey.

Sample Code for Test JSP Header Section Survey Resource

Following is the source code for the sample test JSP header section.*This sample is provided for informational purposes only.* Oracle is not responsible for the correct implementation of JSP in your environment, and does not provide support for customized survey resources. Consult appropriate HTML and JSP experts for more information.

```
<!-- $Header: iessvytestheader.jsp $ -->

<table align=center border=0> <tbody> <tr> <td> </td> <td
class=pageTitletcolspan=4 noWrap>Test Header Section</td>
<td> </td> </tr> </tbody> </table>
```

Survey Campaign Setup Examples

The primary reason to create more than one deployment is to use different lists, since lists are associated at the deployment level. Thus, a cycle with several deployments may be executed over the same time period but with different lists. These deployments would then be executed by different audiences over the same period of time.

The practical reason to create more than one cycle for a single survey campaign is to execute the same deployment (or set of deployments) over a different period of time.

As an example, company ABC decides to conduct a survey campaign to measure customer satisfaction with its products. It wishes to send out the same survey questionnaire six months apart:

- Once before it introduces a new line of products

- Once afterwards

ABC customers are identified by three lists:

- Direct mail customers

- Internet customers
- Retail customers

The survey will be executed in two separate time periods:

- January through March
- July through September

The survey administrators can choose to define two separate cycles (cycle 1 and cycle 2) as child objects of the customer satisfaction survey campaign.

In this model, each cycle represents a given time period for survey execution. Define three deployments (one for each list) for each cycle. Each deployment for cycle 1 has identical start and end dates (January through March), and each deployment for cycle 2 also has identical start and end dates (July through September). All other parameters are identical.

Alternatively, one cycle can be created to contain all deployments. In this model, the first three deployments (one for each list) contain identical start and end dates for the same period, January through March. The last three deployments for this cycle (again, one for each list) use identical start and end dates for the second period of time, July through September.

Survey Administration Console

The Survey Administration console is an HTML administration user interface through which survey campaign administrators can set up, execute, and monitor survey campaigns.

This console consists of four tabs. The tabs and the main functions that you can perform in each of the tabs are as follows:

Tab	Main Functions
Survey Campaigns	Administer survey campaigns, cycles, and deployments. View responses received from existing survey campaigns.
Survey Resources	Administer survey resources, which are HTML or JSP files that appear as header or footer sections, error pages, or final pages for scripts executed in a Web browser.

Tab	Main Functions
Audience	*For targeted survey deployments only:* • Perform list management supporting targeted survey deployments using Oracle Marketing.
Invitations	*For targeted survey deployments:* • Perform management of Oracle One-to-One Fulfillment master documents, queries, and templates, which provide the ability to send through e-mail invitations and reminders.

References

- For administrative procedures regarding setting up survey campaigns, cycles, and deployments, or to view responses received, or for survey resource administration, refer to the Survey Resources Administration Tasks and Survey Campaign Administration Tasks sections of *Oracle Scripting Implementation Guide*.
- For specifics on marketing list administration or fulfillment processing, refer to Oracle Marketing and Oracle One-to-One Fulfillment documentation respectively.

The Survey URL

To execute a script in a Web browser, you must access a specifically constructed survey URL using an appropriate Web browser. Typically, the survey respondent clicks a hypertext link which directs the user to the first page of the survey. For targeted survey deployments, this hypertext link can be embedded in an e-mail message inviting (or reminding) the list member to participate in the Web script or survey.

This active survey URL can also be accessed by clicking a button or image, if the referring HTML page contains a hypertext link to associate the image or button with the proper URL.

This section consists of the following topics:

- References for Survey URL Parameters, page 5-29

Generating a Valid Survey URL

When a survey deployment is activated, the survey administration function of Oracle Scripting generates a URL through which the survey deployment is accessed using the Scripting Engine Web interface.

For standard (non-list-based) surveys or Web scripts, this URL (as generated) can be used to directly execute the survey questionnaire script in an Oracle Applications certified Web browser, as soon as the deployment is activated.

The URL for targeted (list-based) survey deployments is not complete upon survey deployment activation. When the Oracle Marketing list is subsequently processed by Oracle One-to-One Fulfillment, additional URL parameters are concatenated to the system URL for each list member, to identify the unique respondent from the Oracle Marketing list. This complete URL, including respondent ID, is included in each invitation or reminder master document sent from the fulfillment server. If the Maximum Responses Per Person deployment parameter is set to 1 for a targeted deployment, this enforces uniqueness (based on the respondent ID) so that the designated list member can only execute the script in a browser once. For a specific list member, the same respondent ID is used for each instance of an invitation or reminder in a cycle.

The construction of the survey Web URL can differ based on these factors:

- Base technology stack of the parent survey campaign (OA Framework or JTT)
- Deployment type (standard or targeted)
- Hosting options (standalone or menu based)

Survey URL Syntax

The general syntax of the survey URL is:

```
http://<server name>.<domain>:<Apache Web server port>/OA_HTML/<hosted,
standalone, or OA survey JSP template>?<Deployment ID>&<Respondent
ID>&<database connection (DBC) information for current session>
```

Survey URL Parameters

- The server name and domain identify the Web server and organization on an Intranet or the Internet.
- The Apache Web server port identifies a specific Apache web server instance either protected by a corporate firewall or accessible to the networked world at large.
- OA_HTML is the Oracle Applications bin on the Oracle Applications server in

which HTML files are stored for execution.

- The JavaServer Page template specifies a set of criteria used for the execution of the script in a web browser client using the Scripting Engine Web interface. This parameter identifies which base technology stack is used to execute the script at runtime. For JTT deployments, it also indicates the hosting option selected.

  For details of the JSP templates, see JSP Templates for Scripting Engine Web Interface Runtime Execution, page 5-27.

  - If this parameter uses the OA.JSP template, the deployment is created as part of a survey campaign with OA Framework selected as the base technology stack. When the URL is invoked in a Web browser, the script executes in the Web browser using the OA Framework technology stack. A URL for such a deployment might appear at runtime similar to the following:

    ```
    OA.jsp?OAFunc=IES_SURVEY_OARUNTIME
    ```

  - If this parameter uses IESSVYMAIN.JSP or IESSVYMENUBASED.JSP as the template, the deployment is created as part of a survey campaign with Deprecated - JTT selected as the base technology stack. When the URL is invoked in a Web browser, the script will execute in the Web browser at runtime using the Deprecated - JTT technology stack.

  - The JSP used identifies the source of authentication for the Oracle Applications session in which the script is executed. If the base JSP template is IESSVYMAIN.JSP, authentication for the Oracle Scripting session is provided using a guest user. The only function allowed by that guest session is to execute a script in a Web browser, one time, and then the Oracle Applications session is terminated when the final page survey resource or URL is displayed.

  - If the base JSP template is IESSVYMENUBASED.JSP, authentication from an existing Oracle Applications session is used to launch the Oracle Scripting transaction in a Web browser, and the tab-based menus associated with that session are hosted in the survey or Web script user interface. After the final page survey resource or URL is displayed, the applications session is maintained, and the user can access any functionality accessible in the hosted application.

- Deployment ID or dID specifies a unique survey campaign deployment in a specific instance of Oracle Applications.

- Respondent ID or rID is only used for targeted deployments, and identifies a specific list member on an Oracle Marketing list. For standard deployments, this parameter is omitted.

- DBC information specifies the database connection information for an authenticated Oracle Applications session.

Survey URL Parameters for Targeted Survey Deployments

Additional information is also appended to the survey URL for targeted survey deployments only, which makes that information available to the Scripting blackboard during execution of the script.

The blackboard key names for these parameters are IES_FND_USER_NAME, P_PARTY_ID, and P_CONTACT_PARTY_ID. These values, respectively, designate the apps user ID of the current logged in user, and (from TCA) the parent party ID and the contact party ID.

The apps user ID contains the login information for the current Oracle Applications session. The TCA parameters may identify the organization or the individual contact for a selected customer record; if not available, either or both of these values could be null.

For more information, see *Oracle Scripting Implementation Guide*, specifically Integrating Oracle Scripting > Integration by Component > Script Author > Scripting Engine Web Interface > Targeted Deployment Integration with Oracle Marketing and TCA.

Extra URL Parameters

Before Release 12, you could freely add your own user-defined parameters to the URL, such as &customer_id=1234.

Starting with Release 12, changes to the underlying Oracle applications code require that all applications register URL parameters for extra security. By definition, user-defined parameters are not known in advance, and therefore cannot be registered by Oracle Scripting.

You can continue to create and run survey campaign scripts with user-defined parameters, but only if you set the profile option FND Validation Level to None; if the value is set to Error, and the URL contains user-defined parameters, the script will error out.

To enable survey campaign scripts to run in all cases, regardless of the value of FND Validation Level, Oracle Scripting has introduced the following 10 pre-registered, case-sensitive parameter names for you to use as extra parameters: **iesK01, iesK02, ..., iesK09, iesK10**.

At runtime, Oracle Scripting will save the parameter key-value pairs entered on the URL in the Scripting blackboard.

See Also

- Guidelines for Survey URL Parameters, page 5-29

JSP Templates for Scripting Engine Web Interface Runtime Execution

Users of the Scripting Engine Web interface access scripts either as survey questionnaires, or as Web scripts:

- Executing a script as a Web-based survey involves only a single Oracle Scripting transaction (and precludes multiple transactions).
- Web script use cases support multiple Oracle Scripting transactions if required. The number of Oracle interactions allowed, as well as the authentication scheme and method of initiating an Oracle Applications session, is governed by the use of a specific JavaServer page as the survey or web script template.

The template used differs based on two factors: the base technology stack selected for execution at runtime, and the intended authentication for the session. Oracle Scripting currently supports the JSP templates that appear in the following table:

JSP Template	Tech Stack	Authentication	Description
OA.JSP	OA Framework	For standard deployments, guest user authentication or menu-based hosting	• Used for all survey campaigns executed in the Oracle Applications Framework technology stack. • Supports applications guest user login. This limits users to execution of the script in a Web browser only. • Supports menu-based hosting for standard deployments. This requires an authenticated Oracle Applications session. In this case, the menus from the hosting application are shown in the UI when the script is executed.
OA.JSP	OA Framework	For targeted deployments, guest user only.	• Does not support menu-based hosting for *targeted* deployments. • Uses applications guest user only

JSP Template	Tech Stack	Authentication	Description
IESSVYMAIN.JSP	JTT	Guest user only	• Uses applications guest user login. This limits users to execution of the script in a Web browser only. • Used for JTT survey campaigns. • Default for standard deployments.
IESSVYMENUBASED.JSP	JTT	Uses authentication from existing Oracle Applications session only.	• Uses menu-based hosting for standard or targeted deployments. • Targeted deployments for JTT survey campaigns are established by selecting Menubased in the Hosting Options (deployment details). • This requires an authenticated Oracle Applications session. • Used for JTT survey campaigns. • The menus from the hosting application are shown in the UI when the script is executed.

Guidelines for Survey URL Parameters

All surveys are executed at the deployment level, with each deployment in a particular instance (based on Apache port) having a unique deployment identification (dID). All respondents for a given deployment will access the same dID. For standard deployments, the dID is the last parameter in a valid URL.

Targeted (list-based) survey campaign deployments include an additional survey URL parameter: a unique respondent identification (rID) for each list member. Every list member will access the same dID, but will have a unique rID for the valid URL.

References for Survey URL Parameters

For information on adding parameters to the survey URL to pass values into the Oracle

Scripting blackboard for a Scripting Engine Web interface session, see the section Passing Parameters to the Web Interface in *Oracle Scripting Developer's Guide*.

Survey Reports

After summarizing survey data for optimum performance using Concurrent Manager, reports on scripts executed in a Web browser can also be generated using Oracle Business Intelligence.

Custom reports can also be generated from tables in the Oracle Applications schema as desired, using tools such as Oracle Discoverer.

Prototypes

Prototype is a boolean characteristic of survey campaigns. The default for this characteristic is null (survey campaigns are not designated as prototypes unless you select this option when creating or editing a survey campaign).

Prototype survey campaigns are identical to standard survey campaigns, except that the script used as the survey campaign questionnaire is not locked. The purpose is to allow survey campaign administrators more freedom to refine requirements for survey campaigns, including modification to the script for the designated survey campaign.

The Survey Campaigns tab includes an option to exclude prototypes. Selecting this option filters prototypes out of the lists of survey campaigns displayed. This option is also null by default (prototype survey campaigns are included in survey campaign lists unless you select this box and click Go).

Although you can also put a prototype survey campaign into production, the script will not be locked. The prototype option can be selected or cleared while a survey campaign status is Open. Thereafter, you cannot change this option.

Concurrent Programs Supporting Survey Operations

Oracle Scripting uses three seeded concurrent programs to support survey component operations. Concurrent programs are executed using Oracle Concurrent Manager. Seeded concurrent programs can be scheduled to execute at certain times, or can be administered to execute in near-real time.

To schedule or execute concurrent programs supporting Oracle Scripting's survey component, you must access Oracle Forms-based applications with a user assigned the iSurvey User responsibility. This section describes in detail each concurrent program applicable to Oracle Scripting.

This section includes the following topics:

- Summarize Survey Data Concurrent Program, page 5-37
- Update Deployment Status Concurrent Program, page 5-38

References

For information on executing concurrent programs or checking concurrent program logs, see Administering Concurrent Programs for Survey Execution in *Oracle Scripting Implementation Guide.*

SUBMIT GROUP FM REQUEST FROM IES Concurrent Program

When you activate a targeted deployment, a concurrent request is generated which is scheduled to run at the deployment start date and time (specified when defining a deployment). If the deploy start date and time are in the past, the concurrent program executes immediately. The purpose of this initial execution of the concurrent program is to make a request to Oracle One-to-One Fulfillment to send out e-mail invitations to the target population specified by the list.

> **Note:** Immediately upon activating a deployment, when the deployment details page refreshes the corresponding Concurrent Request ID is visible at the bottom of the page. You can use this Concurrent Request ID to monitor the Fulfillment Request submission through the Oracle Forms-based interface associated with the iSurvey User responsibility.

If reminders are also established for this deployment, subsequent concurrent requests are also generated (one per reminder, based on the value entered into the Number of Reminders field) to execute at the specified interval (in days) as defined for the deployment. These requests will execute at the appropriate interval (start date and time, plus the number of days defined in the Reminder Interval In Days field). When the concurrent manager runs the request, it attempts to submit a fulfillment request for this survey deployment.

If the request is successful, a fulfillment request ID will be generated. This is visible from the Survey Administration console in the Deployment Detail page, immediately above the Concurrent Request ID.

After activating a deployment, wait a brief period and refresh the page to view the fulfillment request ID. If this ID is listed, the fulfillment request was successfully submitted. If not, the deployment enters the error status.

Troubleshooting

If the fulfillment request is not successful, you have the following options:

- You can attempt to determine and resolve the problem, and reactivate the

deployment.

- You can attempt to manually run this concurrent program.

Resolving and Reactivating the Deployment

When a request fails, the deployment status changes to error status, and the Activate button is again visible. Before you attempt to reactivate the survey deployment, check the concurrent program log for information about the specific error. (For more information, see the Checking Concurrent Program Logs section of *Oracle Scripting Implementation Guide*.)

- Fix the error, and then return to the Survey Administration console to the detail screen of the relevant deployment.

- Click **Activate** to reactivate the deployment.

If the fulfillment request ID appears following a page refresh, the problem is resolved.

Manually Executing the Concurrent Program

When you activate a deployment, the required parameters are automatically passed to the function. If executing manually, you must manually enter the required parameters.

There are two cases in which this concurrent program needs to submitted manually:

1. If the initial deployment fails.

2. If the reminder concurrent program fails.

Follow the procedure detailed in the Scheduling or Executing Concurrent Programs section of *Oracle Scripting Implementation Guide,* using the parameters for the SUBMIT GROUP FM REQUEST FROM IES concurrent program as listed in the table below.

Parameters 7 and 9 below require specific values only if the purpose for manually executing this request is the failure of a previously scheduled concurrent program for reminders. Initial requests (to send out invitations) do not require these parameters.

No.	Parameter	Meaning	Value
1	p_api_version	PL/SQL API constant required by Apps.	1

No.	Parameter	Meaning	Value
2	p_init_msg_list	Initializes a fresh message list prior to loading new messages. This parameter required by Apps, primarily for error tracking. Leaving this parameter null or entering the value FND_API.G_TRUE passes a value of true (yes, initialize the list); otherwise, pass FND_API.G_FALSE, to ensure the message list is not initialized.	Leave parameter null (this passes a true value)
3	p_commit	Tells the API whether or not to commit changes. If set to false, the concurrent manager commits the database transactions instead of the API. If set to true, the API commits the database transaction.	FND_API.G_FALSE
4	p_validation_level	Parameter required by Apps, to determine which validation level is required. Passing a value of 100 results in the API ensuring full validation by passing FND_API.G_VALID_LEVEL_FULL.	100
5	p_deployment_id	Deployment ID for survey campaign deployment	Determine the deployment ID and use it here

No.	Parameter	Meaning	Value
6	p_template_id	Template ID for the invitation or reminder template used in survey campaign deployment.	Determine the template ID and use it here. f manually submitting the request, if your initial request failed, pass the ID of the invitation template. If a subsequent reminder request failed, pass the template ID of the reminder template.
7	p_reminder_type	Variable to identify the request as a reminder. For initial request (invitation), null is passed. For reminder requests, the value REMINDER is passed.	If your initial request failed, leave the parameter null (there is no reminder for an invitation request). If a subsequent (reminder) request failed, type the value REMINDER.

No.	Parameter	Meaning	Value
8	p_user_id	User ID value derived from the Oracle Applications user account for the user submitting the concurrent request	Determine the USER_ID value for the appropriate Oracle Applications user and provide it here. You need the apps password for this procedure. Steps to obtain this value: • Log into Forms-based applications as a user with the System Administrator responsibility. • Locate the record for the appropriate user. • With focus on the User Name field, examine the field and variable values (Help > Diagnostics > Examine). Enter the apps password when prompted. • In the Block field, enter USER. • In the Field field, search for and select USER_NAME. • The number in the Value field is the value you must enter as the p_user_id parameter.

No.	Parameter	Meaning	Value
9	p_reminder_hst_id	If p_reminder_type parameter is set to reminder, this parameter needs to be set. Otherwise, leave this value null. This is the reminder history ID created by the system when the original deployment is submitted.	If initial deployment concurrent program fails, leave parameter null. If your initial request failed, leave the parameter null (not required for invitation). If a subsequent (reminder) request failed, derive this parameter value by using the following SQL query against the database for your Oracle Scripting environment:select survey_reminder_hst_id from ies_svy_reminder_hst_v , ies_svy_reminders_v where ies_svy_reminders_v.deployment_id = <p_deployment_id> and ies_svy_reminder_hst_v.survey_reminder_id = ies_svy_reminders_v.survey_reminder_id.

Guidance

There are no prerequisites for executing this concurrent program. However, in order for this program to have any useful effect:

- The Survey component of Oracle Scripting must be fully implemented.
- Oracle One-to-One Fulfillment must be fully implemented and configured, and a Fulfillment server must be functional.
- A survey campaign and cycle must already exist, and a targeted deployment already defined.
- The associated list must be valid and accessible to the system.
- A targeted deployment must be active.
- The deployment start date and time, as compared to SYSDATE, must be in the past.
- Fulfillment request status is visible in the Survey Administration console in the deployment detail. However, if you need to log into the Fulfillment Administration console for detailed troubleshooting, you must log into Oracle HTML-based applications with a user that has the One-to-One Fulfillment Administrator responsibility.

References

- For more information on implementing, administering, or using Oracle One-to-One Fulfillment, refer to *Oracle One-to-One Fulfillment Implementation Guide*.
- For specific details on executing or scheduling concurrent programs, please refer to Chapter 5 of *Oracle System Administrator's Guide*.
- See the chapter Troubleshooting Oracle Scripting in the *Oracle Scripting Implementation Guide*.

.

Summarize Survey Data Concurrent Program

When interaction center agents execute a script, or when survey respondents participate in a survey, their individual answers are collected in the Oracle Scripting schema of the applications database.

For generating reports on survey data, information must be moved into summary tables that make the data accessible to reporting tools (Oracle Discoverer workbooks) as part of the Interaction Center Intelligence product family.

Run this concurrent program prior to viewing any reports for the most current data, or schedule this program to execute on a regularly scheduled basis (for example, once daily at an off-peak time, when load on the system is not high).

Parameters

There is a single parameter for the Summarize Survey Data concurrent program. The parameter details are listed in the table below.

No.	Parameter	Meaning	Value
1	p_cycle_id	Cycle ID of the specified cycle for which you want to execute this concurrent program.	Locate the appropriate cycle ID from the p_cycle_id list of values and use it here.

Guidance

There are no prerequisites for executing this concurrent program. However, in order for this program to have any useful effect:

- The Survey component of Oracle Scripting must be fully implemented.
- A survey campaign must already have been created and its children objects (survey

cycle and survey deployment) defined.

- At least one deployment must be active.
- For survey reports, respondents must have participated in the survey. In this way, data is available in the IES schema of the Applications database to summarize for reporting purposes.
- It is not necessary to run this concurrent program to obtain current data for footprinting reports.

Update Deployment Status Concurrent Program

When you define a survey deployment, one set of criteria includes the range of time for which the deployment is valid. This set includes the deployment start date, and the response end date and time.

A concurrent program defined in Oracle Applications for survey operations must be executed on a regular basis to check this response end date and time. This process occurs from the Concurrent Manager and is performed by an individual with the iSurvey User responsibility (typically an administrator or supervisor).

The Update Deployment Status concurrent program compares the response end date for each deployment against SYSDATE, and changes the status of all deployments from active to closed if SYSDATE is past the response end date and time. Additionally, this concurrent program changes the higher level survey campaign status to idle if the survey campaign contains no active or pending deployments.

Oracle recommends executing this concurrent program once daily at an off-peak time (a time when load on the system is not high). This concurrent program can be manually executed. For example, executing this concurrent program could be one of the closing operations an interaction center supervisor performs at the end of the day. Alternatively, this program can be scheduled to run automatically, for example, daily at 2 AM.

Parameters

There are no configurable parameters for the Update Deployment Status concurrent program.

Guidance

There are no prerequisites for executing this concurrent program. However, in order for this program to have any useful effect:

- The Survey component of Oracle Scripting must be fully implemented.
- A survey campaign must already exist and its children objects (survey cycle and

survey deployment) already defined.

- At least one deployment must be active.
- The deployment end date and time, as defined in the deployment detail and compared to SYSDATE, must be in the past.

6

Planning Oracle Scripting Projects

This chapter covers the following topics:

- Introduction
- Planning Agent Interface Projects
- Planning Oracle Scripting Survey Campaigns
- Planning Web Scripts

Introduction

Oracle Scripting includes four components: the Scripting Administration console, the Survey Administration console, Script Author, and the Scripting Engine. Any Oracle Scripting project requires a script to be developed using Script Author. This script can then be executed in either the Scripting Engine agent interface (a Java-based user interface used by agents in the interaction center), or in the Scripting Engine Web interface (executed in an Oracle Applications 12 certified Web browser as a series of JSP pages). The same script can be executed in both interfaces.

Appropriate planning for Scripting projects depends on many factors. Chief among them is the intended runtime execution interface for the script. In other words, will the script be executed in the Scripting Engine agent interface? Typical for this scenario is the use of a script in a call center or interaction center (as a call guide, or to help script interactions between customer service agents in an inbound, outbound, or blended environment). Or will the script be executed in the Scripting Engine Web interface (typical uses include using the script as the survey questionnaire in a survey campaign to gather information from a targeted population, or as a Web script from a self-service Web application such as Oracle iSupport). Different considerations apply, and obviously all considerations are applicable in the case of a script intended to execute in both Scripting Engine user interfaces.

Planning Agent Interface Projects

Planning Scripting Engine agent interface projects is typically the task of a consulting group within an enterprise. This involves gathering detailed requirements for building the script, understanding the business rules, integration requirements, and short- and long-term goals of interaction center managers. Thus, the primary aspects of planning for Scripting project managers involve project scoping and discovery of existing, emerging and future requirements.

> **Note:** Many of these aspects are also required of scripts to be executed as surveys. As such, this information may also prove relevant to scripts to be run in a Web browser.

This section includes the following topics:

Facets of Scripting-Specific Discovery Process

There are three main aspects to the Discovery process as it relates directly to Oracle Scripting:

Text Layer

The text layer refers to the text that is to be read aloud by an agent following a script. In order to appropriately plan a Scripting Engine project, obtain a detailed script or the existing call guide (if one is already in production that will be replaced with Scripting). Ensure you understand fully any changes required to be made to an existing script or call guide.

Logic Layer

The logic layer represents a clear understanding of the logic of the desired script,

including expected flow, criteria for branching into specific functional areas of a script under specified criteria, and ensuring there are no dead ends in flow or inescapable logic loops. In order to create a clear logic layer, it is important that the implementing enterprise and any consultants understand and map out all business rules covering every eventuality. Oracle Scripting is an excellent tool to present information logically and consistently, but the tool alone is not a guarantee of success. A clear understanding of the business requirements is key. Detailed flowcharting of the entire script and all its possible branches should be performed. This may require specifically trained business analysts and must be completed (and agreed upon by both parties) prior to initiating development.

> **Note:** It is to the benefit of enterprises implementing Scripting as much as to the consultants customizing the scripts to have a complete, clear flow and to design and agree upon acceptance criteria based on the logic layer.

Constant business requirements analysis is typically required during script development as the Oracle Scripting technology automates the process of an agent interacting with customers, and replaces any previous technology. Scripting has limitations which are easily overcome when the objective is clear, which is why it is important for the enterprise using this tool to be educated in Scripting's approach, its question-and-answer paradigm, and any specific obstacles that are encountered and overcome during development of the initial script. It is in the enterprise's interest to follow the progress of the development of the initial script to be delivered, so interaction center managers and technical staff can begin to appreciate techniques, approaches, strengths, and so on. In this way, subsequent script development (or modifications to a script that is delivered and accepted by the enterprise) is greatly simplified. This is true whether performed by the enterprise staff alone, or with additional consulting support.

Technology Layer

Considerations for the technology layer include, but are not limited to, an understanding of what technology choices will suit the needs of the script at hand (and specific functionality with in the script). Discovery for the technology layer includes forming choices regarding Script Author objects such as Panels, Groups, Blocks, and appropriate Branches. For example, what technology choice makes for the best, most effective method of text to be delivered by the agent? A single panel? Multiple panels? A group containing logically related questions?

However, the GUI provided by the Script Author has powerful capabilities behind it that greatly extend functionality for the script. Many of these require custom programming. For example, what is the best way for an agent to obtain a set of information required by the business rules? Should the choice be simply to present all customers with a long string of questions in consecutive panels? Or is it better to perform a PL/SQL query to the customer database to determine what information about the customer is known, and determine which remaining information must be collected

by creating complex Java methods that analyze the data placed on the Scripting blackboard by the query, and route the agent only to the appropriate questions?

The technology layer includes consideration of, but is not limited to, the following:

- Creating and implementing Blocks making database calls or using APIs
- Writing PL/SQL procedures
- Writing and storing on the database PL/SQL packages
- Creating custom Java components
- Using APIs to take advantage of Scripting functionality or integrate with other applications
- Creating Shortcuts programmed into Groups on the canvas
- Using Indeterminate branches combined with custom Java methods to perform "jumps" to different functional Groups within the script (uses the Shortcut property of a group, or the panel or block name if the destination object is on a sibling object)
- Populating the global Scripting blackboard with values to be used during execution of the script
- Populating the blackboard with key value pairs (by answering questions in a script session) and retrieving them using custom Java methods to control the script flow
- Creating custom data tracking in custom database tables
- Reusing functionality in existing scripts as templates (importing existing scripts or portions thereof)
- Integrating and modifying best practices scripts or surveys
- Employing reusable commands to incorporate functionality integrating with other Oracle Applications

Bringing Together the Layers

The text layer is typically the first to be considered, and while it may be most important to the enterprise implementing the script, it is the least important implementation consideration, providing that specific text is supplied to those building the script, or that there is some flexibility in modifying the required text (for example when the Script Author approach is best served by breaking up a question into two panels).

The text layer can provide certain challenges. For example, in some cases a proposed call guide or script may have undergone legal review prior to coding with Script Author. In the process of building the flowchart representing the final script or building

the script itself, changes may be necessitated due to gaps in logic or other reasons. This may then require further approval for any changes, by the legal authority or department of an enterprise. This can impact a delivery schedule if not planned for in advance.

The technology layer aspect of the Discovery process is by far the most time- consuming to implement. Nonetheless, success in determining requirements for the technology layer are strongly dependent on flowcharting being provided as discussed in the logic layer section above. Without detailed business analysis, Discovery for the technology layer will be ineffective and implementing this layer more difficult, time- and resource-consuming.

The technology layer includes many hidden tasks and covers integration points with Scripting and other applications. Full analysis of the logic and technology layers may indicate that a project is more manageable and more likely to serve the needs of the enterprise when broken into a phased approach. In this way, enterprises can benefit enormously from benchmarking efforts required in an initial phase, and determine realistic assessments for subsequent phases to add future Scripting functionality. In the meantime, in early phases the enterprise can have the interaction center up and running using Oracle Scripting, taking the opportunity to train agents and staff while experiencing the advantages of the efficiencies and return on investment of automated scripting.

Tools to Aid in Scoping and Discovery

Appropriate planning is crucial to the success of a Scripting implementation. Tools to assist in this planning include the Oracle Application Implementation Methodology (AIM), a methodology that follows the full life cycle of a custom software implementation. Even if Oracle Scripting is only part of a broader implementation of Oracle Applications, it is recommended to plan for it separately, using a separate Scope, Objectives, and Approach (SOA) document (AIM CR.010) to help plan the resources required for this portion of the implementation.

Part of a consultant's skill set is the ability to perform a thorough Discovery process to fully understand an enterprise's requirements, existing infrastructure and environment, and long-term needs. Included below are some tools to aid in scoping a potential Scripting project. These include the Oracle Scripting Discovery Data Worksheet, page 6-6, which helps to gather information specific to Oracle Scripting that should be obtained for the potential Scripting project in general. The other tool provided here is the Oracle Scripting Discovery Checklist Tool, page 6-8, which should be filled out for each distinct functionality expected to be created in a script (for example, each group on the canvas).

Using these as aids to the Discovery process can help gather information that will be crucial to appropriate planning, allocation of resources, and scoping of the effort in general.

Oracle Scripting Discovery Data Worksheet

The following worksheet covers an entire Scripting engagement for which you are attempting to gauge the scope in order to properly provide a time estimate and to assess the skill and resource requirements.

1. For which of the following purposes is Oracle Scripting intended to be used?
   - Pure conversation scripting purposes
   - Desktop integration tool
   - Surveying/Polling
   - Other (describe:)
2. If Oracle Scripting will be used for desktop integration, describe the requirements in general:
3. Characterize the script required:
   - Inbound Call Guide
   - Outbound Call Guide
   - Blended Call Guide
   - Guided Selling
   - Survey Questionnaire
   - Web Script
   - Undetermined
4. Are scripts required that support languages other than English? If yes, identify the language or languages.

   Note that any translation of an existing script must be physically performed with a copy of the script using Script Author. Changing the language global property of a script does not translate panel text, questions, or answer choices.
5. How many individual scripts are required to be developed? If undetermined, please indicate.
6. Is the required script based on an software-driven processes? If so, will any business process reengineering be included as part of this effort? If yes, how many?

7. Is the required flow already flowcharted?

8. Is the required script based on an existing script or call guide? If yes, how many?

9. If required script is based on an existing document, script, or flow, will any business process reengineering be included as part of this effort? If yes, what percentage of the existing script is expected to change?

10. If appropriate, attach printouts of the existing scripts to this document or note electronic file name and format of existing script.

11. What guidance currently exists from which Oracle Scripting scripts will be developed?

    - Narrative
    - Flowcharting
    - System Requirements Document (SRD)
    - Existing scripts or call guides

12. Is Computer-Telephony Integration (CTI) intended for use in the facility or facilities? Does the customer have a need to display different call guides using CTI? If so, what will be the criteria?

    - Dialed Number Information Service (DNIS)
    - Automatic Number Identification (ANI)
    - Unique ID (UUID)
    - Account type
    - IVR information
    - Other (Describe:)

13. Is Interactive Voice Response (IVR) unit information required? If so, is IVR in place?

14. Does CTI currently exist? If so, describe:

15. From which integrated Oracle business application is the script expected to be launched?

    - Oracle TeleSales
    - Oracle TeleService

- Oracle iSupport
- Other (list:)

Note that executing scripts in the Scripting Engine agent interface in standalone mode (without other Oracle Applications) is not supported except for script testing and evaluation.

16. If using Oracle TeleSales, does a marketing campaign already exist in Oracle Marketing?

    These applications require a script to be associated with a campaign schedule which in turn is associated with a specific Oracle Marketing campaign.

17. Is integration with other Oracle Applications required?

18. For survey campaigns, will required survey campaigns be targeted (list-based) or standard (non-list-based)?

19. For targeted campaigns only: Do appropriate lists already exist in Oracle Marketing?

20. For targeted campaigns only: Will invitations only be required, or invitations and reminders?

    Note that the seeded Survey List Query must be used as the basis for the query associated with an invitation or reminder master document.

21. Will staff at the implementing enterprise create and maintain its own scripts?

22. **Gap Analysis.** Describe in detail any modifications that Oracle (or partner) must make to the generic version of Oracle Scripting in order to satisfy the needs of the customer.

Oracle Scripting Discovery Checklist Tool

First, qualify each call guide or script required using the following checklist. Then, break down the requirements for *each distinct functionality within each script* (as represented by groups in the script, or functions separated by agent roles) as the Discovery and Scoping process continues.

Be careful to label each main script checklist, and its dependent checklists, appropriately.

Discovery Checklist

Checklist ID _____

Checklist type:

____ Top-level script checklist ____ Script functionality breakdown checklist

Check	Number of Panels	Check	Development Level Assessment
___	Very Small (1 to 25 panels)	___	Very Simple
___	Small (25 to 40 panels)	___	Simple
___	Medium (40 to 100 panels)	___	Lower Intermediate
___	Medium to Large (100 to 200 panels)	___	Upper Intermediate
___	Large (200 to 400 panels)	___	Complex
___	Very Large (Over 400 panels)	___	Very Complex
	Comments:		Comments:

Check	Required Script Objects	Check	Required Branching Types
___	**Panels**	___	**Very Simple**
___	**Groups**	___	**Simple**
___	**Blocks**	___	**Default**
___	**Primarily Panels and Groups**	___	**Distinct**
___	**All object types**	___	**Conditional**
		___	**Indeterminate**
	Comments:		Comments:

Check	Script Creation and Modification Method	Check	Panel Modification and Customization
___	Graphical script only	___	Primarily use automatic panel layout

Check	Script Creation and Modification Method	Check	Panel Modification and Customization
___	Wizard script only	___	Moderate panel HTML customization
___	Started with wizard, converted to graphical script	___	Substantial panel HTML customization
	Comments:		Comments:

Check	Database Integration	Check	PL/SQL Integration
___	No database will be used	___	No PL/SQL will be used
___	Database integration required	___	Custom PL/SQL commands required (amount)
___	Custom tables required	___	Required for Oracle Applications schema only
___	Custom schema(s) available (if so, attach)	___	Required for custom tables (schema required)
	Comments:		Comments:

Script Author Command Types	Required? (Yes/No)	Resourced? (Yes/No)	Planned? (Yes/No)	Template Exists? (Yes/No)
PL/SQL				
Java				
Blackboard				
Constant				
Forms				

Technology	Required? (Yes/No)	Resourced? (Yes/No)	Planned? (Yes/No)	Exists? (Yes/No)
HTML				
Graphics				
Hyperlinks				
Best Practice Java Methods				Yes
Custom Java Methods				
JavaScript				
Java Beans				

Integration	Required? (Yes/No)	Resourced? (Yes/No)	Planned? (Yes/No)
Oracle TeleSales			
Oracle TelesService			
Oracle Forms			
Oracle iSupport			
Oracle One-to-One Fulfillment			
Oracle Marketing			
Discoverer			
Legacy Application			

Check	Scripting Engine Execution Requirements (Check all that apply)
___	**Agent Interface**
___	Execute scripts in standalone mode (no calling business application)
___	Integrate with Customer Support component of Oracle TeleService
___	Integrate with Oracle TeleSales
___	Integrate with other Oracle Forms-based applications (Identify)_____
___	**Web interface** (requires survey campaign administration)
___	Execute as survey
___	Execute as web-based script
___	Integrate with Oracle iSupport
___	Integrate with Oracle iStore
___	Integrate with other Oracle HTML-based applications (Identify)_____
___	Includes targeted (list-based) deployment
___	Includes standard (non-list-based) deployment
___	List required? (Indicate name if existing)_____ Number of list records: _____
___	Invitation required? (Indicate name if existing)_____
___	Reminder required? (Indicate name if existing)_____
___	Query required? (Indicate name if existing)_____ (Must be based on Survey List Query)

Planning Oracle Scripting Survey Campaigns

The main purpose for executing scripts as surveys is to obtain data (via a clearly defined survey questionnaire) from a population over a specific period of time for a particular business purpose. Using a script as the questionnaire, enterprises can rapidly solicit and receive such data at low cost by defining a survey campaign in the Survey Administration console. This data is then typically collected and analyzed to serve the enterprise by improving products or processes, or otherwise allowing the enterprise to be responsive to its polled population. Using an Oracle Scripting-specific end user layer in Oracle Discoverer, survey administrators or interaction center managers can subsequently create on-demand reports based on the data received from survey respondents.

This survey data may be solicited from a targeted population (using predefined Oracle Marketing lists) or from a general population. Targeted populations may include customers or prospective customers, but also partners or affiliates, a company's own employees, and so forth. Business purposes may in some cases be served by responses from a random population. At other times business purposes may be best served by receiving feedback from an identified population. Both are supported by Scripting Engine Web interface.

Summary of Process Flow Steps

Step	Description
1. Define survey campaign requirements, page 6-15	Obtain requirements for all aspects of the survey campaign (survey campaign, cycle, and deployment-specific requirements). These should be documented using appropriate methodology; for example, if using the Application Implementation Methodology (AIM), obtain Scope, Objectives and Approach document and Business Requirements documents such as BR.100.
2. Create and deploy script, page 6-19	Create script for survey questionnaire based on documented requirements (captured in step 1) and deploy to appropriate applications database.
3. Decision: Targeted deployments in survey campaign, page 6-24	Are deployments in the survey campaign targeted (list-based), standard, or both? If targeted deployments = NO, go to step 5. If YES, continue at step 4.

Step	Description
4. Generate lists (for targeted survey campaigns), page 6-25	Generate, create or import lists in the Survey Administration console (one list per deployment, if multiple deployments are required). Note that the same list can be used for multiple deployments (and also for multiple survey campaigns). Generating lists requires the implementation of list management portions of Oracle Marketing. Note that successful delivery of invitations (and reminders) to participate in survey campaigns requires fully implemented and configured Oracle One-to-One Fulfillment (including definition of an outgoing e-mail server) as well as Oracle Marketing.
5. Administer survey resources and survey campaign details, page 6-26	Create and upload (or modify existing) survey resources to be displayed when using survey campaigns. Then define these survey resources in the Survey Administration console. The preceding comprise all survey resources administration steps. Subsequently, administer the survey campaign details in the Survey Administration console. This includes creating a survey campaign and cycle.
6. Define deployments, page 6-27	In the Survey Administration console, define deployment information for one or more deployments subordinate to each cycle in a survey campaign, based on documented requirements. For targeted campaigns, this will include lists, invitations, and (if included in requirements) reminders.
7. Activate survey deployment, page 6-28	In the Survey Administration console, set deployments to Active status. This makes the survey campaign available to respondents immediately (or, for targeted deployments, as soon as the invitations are delivered by the fulfillment engine).
8. Monitor ongoing results, page 6-30	Monitor survey campaign. From the Responses tab of the Survey Administration console, individual responses can be monitored from the moment a deployment becomes active. Each question and the response selected is displayed per respondent.
9. Collect information in Oracle RDBMS, page 6-30	As respondents complete survey questionnaires, information is collected in Oracle RDBMS. No action is required to collect this data.
10. Report survey results, page 6-30	Using Oracle Discoverer along with survey-specific concurrent programs, generate on-demand reports using existing Discoverer workbooks. Knowledgeable Oracle Discoverer administrators can also generate custom reports. Requires implementation of an Oracle Scripting end user layer.

Planning

This section includes the following topics:

Gathering All Survey Campaign Requirements

Prior to creating a script to use as the survey questionnaire or to administering survey campaign data in the Survey Administration console, a survey campaign must be planned, from top-level strategic aspects down to a detailed level. This includes gathering the requirements for all aspects of the survey campaign, from top-level goals to the names of cycles and deployments and start and end dates of each deployment. Step 1 of the business process flow includes the determination of information that is required for all other steps in the business process flow.

Planning Requirements for All Survey Campaigns

Substantial information must be gathered in the planning stage prior to proceeding to step 2 of the business process flow or beyond. If you are using AIM in your implementation, this data would be identified in various AIM documents.

The following information is required for *all survey campaigns:*

- The goals of a survey campaign must be clearly documented.
- The target population (if any) must be identified.
- The method of obtaining survey data (the respondent entry point) must be determined.

- The requirement for targeted or standard deployments must be indicated. Standard deployments have substantially fewer requirements.
- Survey questionnaire requirements must be obtained to a detailed level. The manner in which data will be collected must be defined, including data type, collection method, sequencing and flow.
    - What type of data will be collected from respondents?
    - How will this data be used or evaluated?
    - Will branching logic will be employed? Or will the same questions be asked of all respondents, regardless of previous answers supplied by respondent?
    - What types of technologies (Java, PL/SQL, Oracle Forms, and so on) will be included in the survey questionnaire script?
    - What level of skill is required to incorporate the appropriate technology into the script?
- In order to ensure that the business goals of the survey campaign will be met, a detailed flowchart and script test plan must be planned for.
- The layout, configuration and composition of survey resources (header section, footer section, error page, and final page, if used) intended to display to Web script or survey users must be identified.
- The definition of survey resources must be accomplished prior to defining the survey campaign.
- The survey campaign name must be identified.
- Optionally, a description of the survey campaign (generally including its intended purpose) can be provided.
- The base technology stack in which the script is to be executed in a Web browser must be determined. Oracle recommends using the Oracle Applications Framework technology stack, unless compelling reasons exist for using the deprecated JTT technology stack at runtime.
- Designation of a survey campaign as a prototype must be identified in advance. This option cannot be changed once the survey campaign is saved.
- The survey questionnaire script name (the name designated in the Script Author file properties) must be identified.
- Name and number of cycles required must be identified.

- Deployment name for each deployment must be identified.
- Number and type of deployments per cycle must be identified. Deployments can be either targeted (list-based) or standard (non-list-based).
- Deployment start date and time must be identified per deployment.
- Deployment response end date and time must be identified per deployment.

Additional Planning Requirements for Targeted Deployments

For targeted (list-based) deployments, the following must be considered:

- The hosting option, which determines authentication method at runtime.
- Enterprise Web server and port must be identified as parameters to construct the survey URL, if different than the environment in which administrators log in.
- Oracle Marketing and Oracle One-to-One Fulfillment must be implemented. A fulfillment mail server (typically Oracle Email Server) must be configured, and a functioning fulfillment engine established.
- Oracle Marketing list must exist, the name must be known, and the list accessible.
- Business rules regarding maximum number of responses per Web-based script user must be identified. When set to one (the default value), a list member can take a survey or execute a Web script only one time.
- The target response percentage for survey cycles must be identified.
- Invitation template must exist and be known, including a master document and associated query. Alternatively, these can be created by a skilled fulfillment administrator at the time of deployment definition (survey flow step 6).
- E-mail invitation message subject must be identified.
- Requirement for reminders must be determined. If required, Reminder template must exist and be known, including a master document and associated query. Alternatively, these can be created by a skilled fulfillment administrator at the time of deployment definition (survey flow step 6).
- The number of reminders and interval between (in days) must be identified if reminders are to be employed.

Methodology

Survey campaign requirements and detailed information should be collected and documented using a consistent methodology.

Application Implementation Method

Oracle customers, partners and consultants have access to the Application

Implementation Method Advantage (AIM). AIM is a set of pre-packaged approaches for implementing Oracle Applications, developed by Oracle's Applications Global Service Line group. There are two subsets (AIM Advantage and AIM FastForward). Each is a proven, scalable toolkit for implementing Oracle Applications.

AIM is broken into six phases to cover the full software life cycle: Definition, Operations Analysis, Solution Design, Build, Transition, and Production.

For the various phases, AIM provides macro-driven document templates to address numerous processes, which typically span more than one phase. Each of these processes is described by a two-letter acronym, as described in the table below.

Acronym	AIM Process
CR	Customer Requirements (Project Management)
BP	Business Process Architecture
BR	Business Requirements Definition
RD	Business Requirements Mapping
TA	Application and Technical Architecture
MD	Module Design and Build
CV	Data Conversion
DO	Documentation
TE	Business System Testing
PT	Performance Testing
AP	Adoption and Learning
PM	Production Migration

Documents generated for each AIM process are identified by the process acronym and a number. For example, the top-level document describing the scope, objectives, and requirements of a project (the main customer requirements) is referred to as a CR.010. You may see specific documents within the AIM methodology referenced in various Oracle documentation.

Note that AIM is *not* a requirement for customer implementations. It is one example of a

proven methodology. While planning is required in order to execute an implementation and the subsequent use and administration of a system successfully, *any* effective methodology will suffice.

Creating and Deploying a Survey Questionnaire

A survey campaign has a one-to-one correspondence with a single survey questionnaire. In other words, each survey campaign employs only one script. These survey questionnaire scripts can only be created, modified, and deployed from Script Author.

Part of documenting the requirements for a survey campaign is the detailed definition of the requirements for the survey questionnaire script. This includes information such as specific questions, data format of responses, validation requirements, target audience, branching logic requirements, supporting technologies, questionnaire sequencing and flow. In a best-case scenario, you will have a detailed flow chart depicting the flow of the script, branching, and so forth. You should also have a detailed script test plan to ensure the script, as designed, meets the requirements of the survey campaign. Before creating the script you will need a full understanding of the campaign and enterprise business rules, dependencies, and any integration requirements.

Following a gathering of the requirements, you will need the following to create, modify, and deploy the survey questionnaire script:

- Appropriate Network, Security, Hardware, and Software Resources, page 6-19
- Trained Script Developers, page 6-21
- Trained Java, PL/SQL, Oracle Forms, and API Programmers , page 6-21
- Trained Oracle Applications and Database Administrators, page 6-21
- Detailed Script-Specific Information, page 6-22
- Environment-Specific Information, page 6-22
- Detailed Survey Questionnaire Requirements, page 6-22

Appropriate Network, Security, Hardware, and Software Resources

The Script Author component of Oracle Scripting is required to build scripts to serve as the survey questionnaire. This requires a networked script development workstation. Network, security, hardware and software resources are indicated below. For more information regarding what is required and recommended for use as a script development workstation, refer to the section entitled "Using Oracle Scripting."

Network

Script Author workstation must be on a network and able to connect with the applications database server in order to deploy scripts.

Security

- Script Author uses the thin JDBC driver to communicate with the database. As of release 11.5.6, the Apache mid-tier architecture enables the Scripting Engine to be executed through a firewall using the HyperText Transfer Protocol (HTTP).

- If the enterprise uses HTTPS (HyperText Transfer Protocol, Secure), then Oracle Scripting must be configured for HTTPS also.

- If the enterprise uses proxy servers, then Oracle Scripting must be configured for the proxy servers also.

- The Network access setting in the Basic tab of the JInitiator Control Panel should be set to Applet Host.

  Note: If using the Caching Architecture of Oracle Scripting only (supported *only* for pre-11.5.6 implementations), *and* if the database and applications servers are on two different physical hosts, the Network access setting in the Basic tab of the JInitiator Control Panel should be set to Unrestricted.

Appropriately configured security settings are the responsibility of Web server administrators at the enterprise.

Hardware

Hardware required to create and deploy a survey questionnaire script (using Script Author) follows the Oracle CRM hardware requirement standards. Any workstation in your enterprise that meets the requirements for running Forms-based applications is suitable.

It is from this workstation that the script will be deployed to the applications database. The script can be created on this workstation or created on another workstation with Script Author, and copied to the workstation within the enterprise firewall to deploy.

Software

Software for Developing Scripts

- Operating system: Any operating system supporting Oracle Applications.

- Script Author Java applet, accessible from Oracle Applications for a user with the Scripting Administrator responsibility.

- Oracle Applications certified Web browser (see Oracle*MetaLink* for description of the latest current certifications).

Software for Additional Development Tasks

Additional software required for Java development:

- Java IDE such as Oracle JDeveloper

- Java Development Kit (JDK)

Additional software required for database and PL/SQL development:

- Appropriately configured TNSNAMES.ORA and SQLNET.ORA files
- SQL tools such as SQL Plus or SQL Worksheet
- Oracle Client (recommended)

Trained Script Developers

Since survey questionnaires are often a series of simple questions and answers, the questionnaire script can be created by relatively non-technical users trained in the use of Script Author. This is particularly true with survey questionnaires employing primarily sequential logic, or building scripts using the Script Wizard.

However, because the survey functionality is based on Oracle Scripting, it also provides the ability to include very sophisticated functions in the script, if required. Therefore, in order to create adequate survey scripts, script developers must have received training in using Script Author.

Based on the complexity of the script in question, script developers must also be knowledgeable in the various supporting technologies, or have access to resources with such knowledge. The selection of supporting technologies required depends on specific requirements for each survey campaign, as well as the degree to which Scripting-supported technologies will be utilized. These technologies can include custom Java and application of APIs, PL/SQL, Oracle Forms, or Scripting-specific commands such as Constant and Blackboard commands. For more information, see Trained Java, PL/SQL, Oracle Forms, and API Programmers, page 6-21 below.

Trained Java, PL/SQL, Oracle Forms, and API Programmers

Enterprises that intend to leverage the full capabilities of Oracle Scripting in survey campaigns may require the services of highly trained Java developers with experience customizing application program interfaces (APIs), PL/SQL programmers, developers experienced in accessing and working with Oracle Forms, logic and flow experts, business process engineers, and so forth.

Use of custom Java will also require deployment of custom code to the enterprise server, which requires an understanding of packaging of JAR files.

Trained Oracle Applications and Database Administrators

Any custom PL/SQL packages for use with the survey campaign must be appropriately built, deployed to the enterprise database, and tested. Deploying PL/SQL packages to the enterprise's applications database may require database administrator (DBA) privileges.

Detailed Script-Specific Information

To develop a script that meets the survey campaign requirements, individuals creating scripts or supporting code will need access to existing flow charts or other documented script-specific requirements. Script developers must also be apprised of any dependencies, business rules, and predefined integration requirements. Upon completing development, script developers will need access to any existing test plans ensuring survey questionnaire requirements have been met.

From an AIM perspective, in order to achieve acceptance of the script it must meet the requirements of the system design (as represented by MD.070 or MD.080) and any and all test documents (as represented typically by the TE.040).

Environment-Specific Information

Scripts must be deployed to the applications database. Using the Script Author Java applet, this can now be performed through a firewall. Scripts are deployed to the specified database instance from the Script Author applet without need to define connection information; previously, one needed to define the database host name, TNS port and SID.

The Script Author Java applet is launched from the Home tab of the Scripting Administration console. Upon clicking Launch Script Author, Oracle JInitiator opens in a new browser window, and the Script Author applet connects to an Apache mid-tier servlet via HTTP. This Script Author servlet makes a JDBC connection to the database, and all database-based operations in Script Author are executed by this servlet. The servlet uses values identified in the Apps Servlet Agent system profile.

To deploy a script to a different environment, you must begin a new Oracle Applications session in the appropriate instance, open the Scripting Administration console, and launch Script Author using the servlet for that instance, and deploy the script. Thus, environment-specific information is now determined entirely by the current Oracle Applications session.

Detailed Survey Questionnaire Requirements

Trained individuals use the Script Author tool to visually lay out the flow of a script based on the script requirements. All of the requirements must be made known to the developer of the script to create an effective survey questionnaire. The information in the table below includes the type of information required to successfully create a script that meets the survey campaign requirements. For any script, other information may be required that is not listed below.

Requirement Type	Description
Panel content	Text (and, for graphical script users, graphics) that appear in any HTML page. This includes any information assigned to the "Label for Reporting" field when defining a panel answer.
Lookup values	Predefined answer choices.
Constant commands (graphical script) or Default Answers (wizard scripts)	Used to provide *or change* an answer default. When executing scripts in a Web browser (using the Scripting Engine Web interface), the first selection in the range of answer choices becomes the default choice. This is different in the Java client, or the Scripting Engine agent interface. In a graphical script, if you wish to provide a different answer choice as the default, use a constant command to provide a different default value for a dropdown, radio button, text field or text area. If using the Script Wizard, you can provide a default value in the Define Question Detail window, when defining a question control of a text, text area, or password type. When defining answer choices in a wizard script for a radio button or dropdown list using the Define Answer Choice for (Panel Name) window, you can specify whether any given answer choice is the default using the **Should this be the default answer for this question** option. When defining answer choices in a wizard script for a multi-select control type (such as the multi-select list or checkbox group) using the Define Answer Choice for (Panel Name) window, you can specify whether any given answer choice is the default using the **Default answer** option. In a wizard script, there is no equivalent to the single checkbox control type. You can use the checkbox group control, in which you have the option of setting the default answer at runtime. There is no method for establishing no answer choice as the default.

Requirement Type	Description
Validation ranges or requirements	Validation can be enforced in a question UI control. For graphical scripts, this requires associating a Java method with a question in the data dictionary and must be explicitly programmed. The requirement for validation on a particular answer (and the range of valid answers, if part of the validation routine) must be provided to the developer of the script in advance. For wizard scripts, validation that a response is provided or validation of a date in a certain range is established by selecting options in the UI.
Branching logic	If a survey questionnaire is appropriately planned, branching logic can be clearly indicated in a flowchart.
PL/SQL packages	PL/SQL commands that are loaded in the applications database can be referenced from a graphical script. All such commands must be identified prior to development of the script. If using the Script Wizard, you must graph a script to provide custom commands.
Database table or view field names and locations	If tables are referenced from a script, the precise table names and locations must be made available to the individual building that portion of the script. If using the Script Wizard, you must graph a script to provide data dictionary details such as tables or views.

Determining If Survey Campaign Requires Targeted Deployments

The third step in the Survey business process flow is a decision block, indicating the requirement to determine if the survey campaign contains requirements for targeted (list-based) deployments. This determination affects the remaining process flow in terms of order of steps and complexity of survey campaign setup and administration.

A survey campaign may have an existing list comprised of members of a targeted population. For a list-based survey campaign, this list is created using Oracle Marketing.

Targeted survey campaigns obtain survey feedback by inviting list members to participate in the survey. This *invitation* is an Oracle One-to-One Fulfillment document that is sent to list members by electronic mail using Oracle One-to-One Fulfillment capabilities. The same list may be used to send *reminders,* an e-mail message typically reinforcing the deadline for list member participation. List members that respond by taking a survey are the survey respondents.

This section includes the following topics:

List-Based Campaign = NO

If the ability to execute targeted deployments within a survey campaigns is not required, step 4 of the process flow (generate lists) is skipped and step 6 (define deployments) is substantially simplified. Prerequisites such as implementation of Oracle One-to-One Fulfillment and Oracle Marketing are also precluded. The abilities you sacrifice by choosing a standard (*non*-list-based) survey campaign include the ability to invite members of a list to participate in a survey, the ability to track survey respondents by name or ID number, the ability to send reminders, or the ability to execute subsequent deployments that reaches the same precise audience.

List-Based Campaign = YES

When planning to implement or use the survey functionality of Oracle Scripting with targeted survey campaign deployments, the business process flow requires list generation as the next step. Some information is included in *Oracle Scripting Implementation Guide.* For more information, refer to Oracle Marketing product documentation.

Generating Lists

Step 4 of the survey process flow is the generation of lists using Oracle Marketing. These lists are used to send invitations and reminders via e-mail to list members, encouraging them to click the included URL and participate in a survey.

Additional Implementation Requirements for Targeted Deployments

Additional implementation requirements for enterprises using targeted (list-based) survey campaign deployments. These require administration of Oracle Marketing and Oracle One-to-One Fulfillment for list management and fulfillment template management, respectively.

Oracle Marketing

Implementation of list management aspects of Oracle Marketing are required. Once these prerequisites are met:

- Oracle Marketing can be used to create and administer lists for use with surveys.
- Lists can be generated (Oracle Marketing functionality) from within the Survey Administration console.

References

See *Oracle Marketing Implementation Guide* for more information on implementing Oracle Marketing. See *Oracle Marketing User Guide* for more information on working with and generating Oracle Marketing lists.

Additional Oracle One-to-One Fulfillment implementation and configuration requirements are prerequisite for executing list-based survey campaigns. This includes:

- Implementation and configuration of the Oracle One-to-One Fulfillment server, including:
  - An identified outgoing e-mail server
  - An associated survey group
  - Survey administrator users added to fulfillment server group
- A functioning fulfillment engine

Once these prerequisites are met, Oracle One-to-One Fulfillment can take Oracle Marketing lists and merge data from those lists into survey campaign-specific invitations and reminders and send these out through the identified e-mail server. These steps must be accomplished by a fulfillment administrator (an Oracle Applications user with the One-to-One Fulfillment Administrator responsibility).

Administering Survey Resources, Survey Campaign and Cycle Details

After obtaining requirements (step 1), creating and deploying a script (step 2), determining whether lists are required (step 3) and generating lists if appropriate (step 4), you are ready to administer survey resources and administer survey campaign information. Survey resources are files that can display when a respondent takes a survey or a user executes a Web script. There are two types of resources: section resources and page resources.

Section resources include the header section and the footer section.

- For scripts executed in the OAF technology stack, these resources are comprised of HTML and are optional. If not designated, no section resource displays.
- For scripts executed in the JTT technology stack, header resources are required, and the footer section is optional. Both must be JSP files.

Page resources include the error page, and the final page.

- For scripts executed in the OAF technology stack, page resources are comprised of HTML. Defining these and specifying a predefined error or final page resource is optional, but the component is not. If not designated, a default error or final page displays at runtime.

- For scripts executed in the OAF technology stack, you can also specify a URL redirect for a final page. This routes the user to the URL you specify upon completion of the script.

- For scripts executed in the JTT technology stack, error and final pages are mandatory. There is no system-generated default. Both must be JSP files.

This information is entered by a Survey administrator in the Survey Administration console, accessed from the Oracle Applications HTML login.

This step in the flow also includes the creation of a survey campaign and its child object, the cycle. Both can be created from the Survey Campaigns tab. A survey campaign may contain more than one cycle. Cycles can also be created from the Survey Campaign**s** tab.

> **Note:** The order of steps described here is not arbitrary. Survey resources must be defined before they are specified.

For planning purposes, the following types of information are required:

Survey Campaign-Level Information

- Script name
- Survey campaign name
- Survey resources:
  - Names for header section, footer section, error page, or final page resources. To use specific resources, they must first be defined.
- Cycle names
- Number of cycles required
- Minimum response percentage

Defining Deployments

Deployments are associated with a specific cycle, which in turn is associated with a single survey campaign. On the same HTML page, information specific to list-based survey campaigns is also required.

> **Note:** If a survey campaign is standard, no list information will appear on the page.

The information you will need in order to define a deployment includes:

Deployment-Specific Information

- Survey under which this deployment is associated.
- Cycle under which this deployment is associated.
- Deployment name for each deployment.
- Deployment start date and time.
- Response end date and time.
- Deployment type (standard or targeted).
- Enterprise Web server URL and port

Targeted Deployment-Specific Information

- The hosting option
- Enterprise Web server and port, if different than default
- Marketing list name
- Maximum number of responses
- Target response percentage
- Invitation template name
- E-mail subject name for invitation
- Reminder template name
- E-mail subject name for reminders
- Reminder interval, in days

Activating Survey Campaigns

Once one or more deployments have been defined under a cycle for a survey campaign, each can be activated immediately.

Standard Deployment Scenario

For standard deployments, the act of setting a deployment to Active status (formerly known as deploying a deployment) enables respondents to take surveys as soon as you press the Deploy button.

Targeted Deployment Scenario

For targeted deployments, setting a deployment to Active status allows e-mail invitations to be sent out based on the date and time parameters established for each specific deployment. This requires:

- An invitation Master Document, with an appropriate query matching all merge fields and associated with the Master Document.
- An Oracle One-to-One Fulfillment template, associating a specific Master Document and its components.
- Another template is required if reminders are used.
- Fully configured lists, typically generated in Oracle Marketing.
- Appropriately configured concurrent processes.

When a deployment is set to Active, a concurrent job request is posted to the Concurrent Manager. The Concurrent Manager is functionality built on Oracle Application Object Library (AOL) classes, which are included in Oracle Applications 12 with an appropriate Rapid Install.

The Deploy Date and Deploy Time are passed to the Concurrent Manager as the appropriate execution time to run the process. If the date and time specified are past the SYSDATE, the job executes immediately. Otherwise, the request remains on the concurrent request queue until the execution time arrives, and a fulfillment request ID. Upon execution time, the fulfillment server is notified to follow the instructions provided to it by the Oracle One-to-One Fulfillment template. The template is associated with a particular Master Document (invitation or reminder) and query. The query pulls the contact information of the list member from Oracle Marketing and unique id from IES_SVY_LIST_ENTRIES to merge them in the Master Document, personalizing the email invitation and providing the appropriate list-based unique URL.

This personalized message, containing the data from all merge fields (at minimum, typically, the customer name and survey URL), is then sent to the outgoing e-mail server identified by the fulfillment group and its member users (Survey administrators), and delivered through the fulfillment server.

If reminders are associated with a deployment, then upon setting the deployment to active, concurrent jobs are submitted for reminders with calculated reminder dates and times, and instructions regarding the date to execute the job. These are also processed by the specified template, merging information as necessary and sending out the e-mail messages to the e-mail server identified with the fulfillment group and its member users.

Monitoring Survey Results

There are no hard and fast requirements for monitoring survey results. As soon as respondents complete a survey over the Web, the information is passed to the Scripting schema in the Oracle applications database. From the Survey Administration console, each individual response can be reviewed on an ongoing basis by accessing the Response View option from the update deployment page of any specific deployment in the Survey Administration console.

Collecting Survey Results in Oracle RDBMS

When scripts are executed, information is automatically collected into the Scripting schema in the Oracle applications database. When surveys are taken over the Web, additional information is made available in survey transaction tables. No action or planning is required for this collection to take place.

Reporting Survey Campaign Deployment Results

If you want to view reports generated by survey operations, you can implement an end user layer (EUL) for the Oracle Discoverer tool.

For more information, refer to Appendix C of *Oracle Scripting Implementation Guide*.

If no surveys in a deployment have received responses, no information will be available to display in these reports.

To generate meaningful data, the Summarize Survey Data concurrent program should be executed to ensure all data in the summary tables have been compiled and are available to view as reports. For information on concurrent programs, see Administering Concurrent Programs for Survey Execution in *Oracle Scripting Implementation Guide*.

From Where Does Reporting Information Derive?

When a script is executed as a survey through the Web, the Summarize Survey Data concurrent program moves survey transaction information from the Scripting schema into survey transaction summary tables from which Survey campaign reports are generated. Summary tables can also be updated at scheduled times, as described in Administering Concurrent Programs for Survey Execution in *Oracle Scripting Implementation Guide.*

For more information, see Reporting and Analyzing Survey Respondent Data in *Oracle Scripting Implementation Guide*.

Planning Web Scripts

Web scripts are scripts executed by users of Oracle self-service Web applications such as Oracle iSupport or Oracle iStore.

In terms of planning, essentially the same level of thought is required for a Web script as is required of a script to be executed as a survey. Since this scenario involves an existing user executing a script from within the application UI, list management and fulfillment template management are not required.

Setup for the purposes of user authentication are also not required, as this scenario uses the Oracle Applications authentication information from the current applications session.

One unique concern is the customization of the Web application user interface to include a valid survey URL, allowing application users to launch a Web script from the application.

7

Using the Script Wizard

This chapter covers the following topics:

- Introduction
- Wizard Script Hierarchy
- Overview of Wizard Script Pages and Operation Flow
- Overview of Standard Wizard Script Setup
- Wizard Pages
- Accessing the Script Wizard
- Managing Scripts in the Script Wizard
- Managing Panels in the Script Wizard
- Managing Questions in the Script Wizard
- Managing Answer Choices in the Script Wizard
- Saving and Deploying Wizard Scripts
- Determining Flow with the Script Wizard
- Wizard Script Flow Strategies
- What Do Wizard Scripts Contain or Exclude?

Introduction

Scripts created by the Script Wizard are referred to as *wizard scripts*. Wizard scripts incorporate a small set of the options available to graphical scripts, and can be useful in starting off a scripting project.

Conversion to Graphical Scripts

Although it is possible to create and deploy wizard scripts, in a typical scripting project, you would use a wizard script as a stepping-stone to creating a fully functional script,

as follows:

- Create a wizard script.
- Graph the wizard script, that is, convert a copy of the wizard script into a graphical script.
- Add further functionality to the graphical script.
- Deploy the graphical script.

> **Note:** You can get a graphical script from a wizard script, but there is no reverse operation; features added to a graphical script cannot be retrofitted into a wizard script.

For more considerations about wizard and graphical scripts, see What Do Wizard Scripts Contain or Exclude?, page 7-25

Wizard Script Hierarchy

A wizard script contains both objects that you define, and some objects that the system attaches automatically (such as Continue buttons on each panel).

This section concentrates on the user-defined objects; the system-added objects are described where appropriate.

The user-defined objects in a wizard script are constructed according to the following top-to-bottom hierarchy:

1. Script
2. Panel
3. Question
4. Answer choice

A wizard script consists of one or more panels. Each panel belongs to one wizard script.

A panel usually, but not necessarily, consists of one or more questions. Each question belongs to a panel.

A question has one or more answers. In defining the question and answers, you decide which one of the following applies:

- The question requires a single answer to be provided: script users must enter text, such as supplying their name or password.
- The question requires a single answer to be selected from a pre-defined list: script

users select either a radio button option or a value from a dropdown list of values.

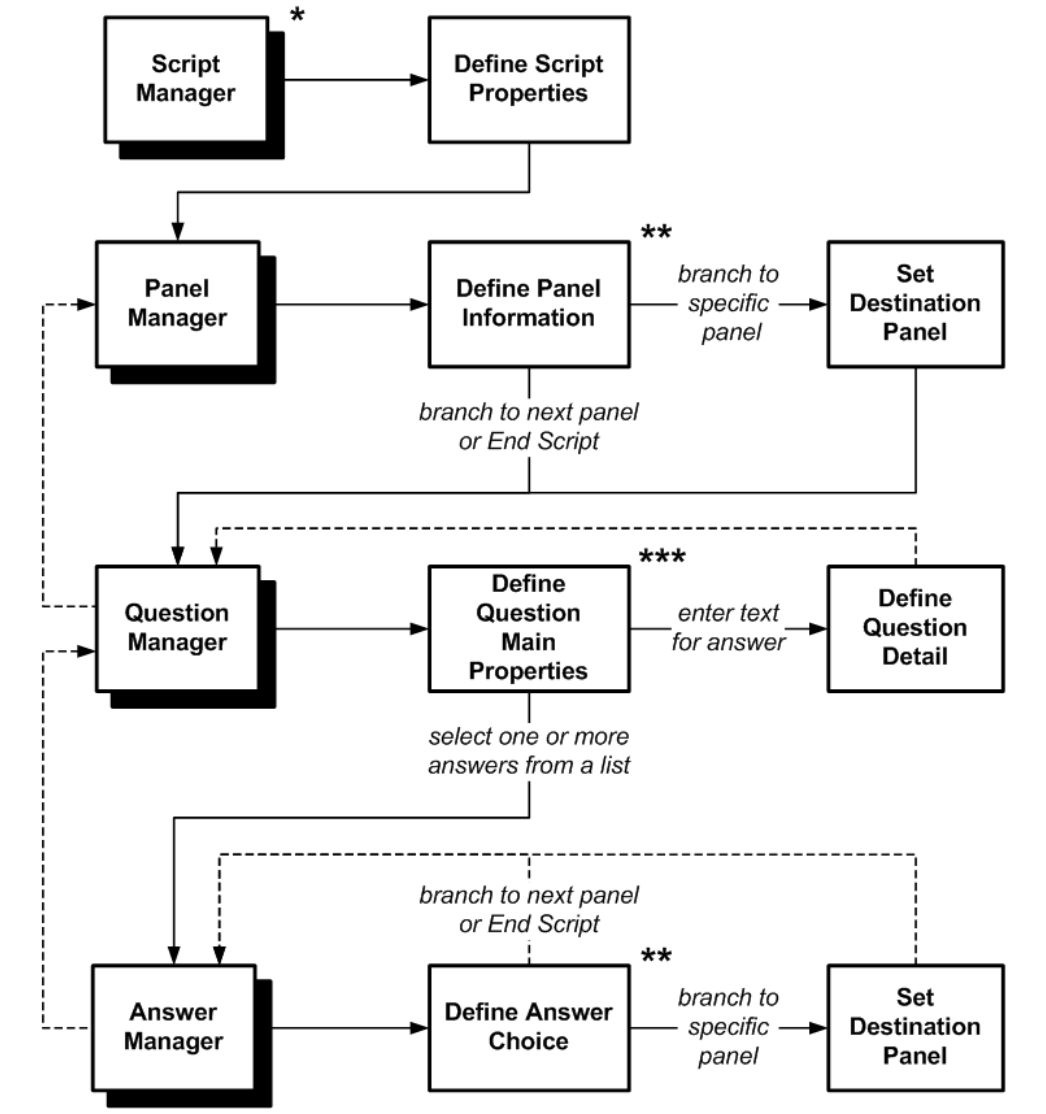
- The question allows none, one, or more answers to be selected from a pre-defined list: script users select either from a checkbox group or from a multi-select list options.

Where the question requires a single, text-entry answer, this requirement is treated as a "question detail" in the Wizard, and you do not define any answer choices.

Where the question requires or allows answers from a pre-defined list, you explicitly define answer choices for the question; these provide the values for the list. Each answer choice belongs solely to its related question.

Overview of Wizard Script Pages and Operation Flow

The following diagram shows the main pages and the general operation flow in the Wizard when you create or edit a wizard script.



Key:

* *In this page, you can graph and deploy the wizard script*

** *The next wizard page depends on whether or not you want to branch to a specific panel*

*** *The next wizard page depends on whether or not text entry is required for the answer*

Notes on the Wizard Pages and Operations Flow

- Each of the "Manager" pages displays a list of the script objects being managed. From the Manager pages, you choose whether to create new objects, or work with existing objects, for example, to edit, copy, or delete them.

- The other, non-Manager pages are where you enter or select details of the wizard

script objects.

For more details, see Wizard Pages, page 7-5.

- You can either create all the objects for a wizard script in one session, or you can save and enter details in several sessions.

Overview of Standard Wizard Script Setup

The standard operations of creating and editing wizard scripts and wizard script objects correspond closely to the Wizard object hierarchy:

A typical way of setting up a wizard script has the following sequence of steps:

1. You create a new wizard script.
2. You create a panel, then generally you create one or more questions, and for the appropriate types of question, the answer choices for each question.

   Note: In the Wizard, panels do not have to have questions.

3. You create further panels, each with their own questions and answer choices as required.
4. Finally, you can either graph the wizard script - to add in extra features - or you can deploy the wizard script.

You can also take a copy of an existing wizard script, and adapt it to your requirements, following the same standard hierarchical flow.

Wizard Pages

This section contains details about the following wizard pages:

- Script Manager, page 7-6
- Define Script Properties Page, page 7-6
- Panel Manager, page 7-6
- Define Panel Information Page, page 7-7
- Set Destination Panel Page, page 7-8
- Question Manager, page 7-8
- Define Question Main Properties Page, page 7-9

- Define Question Detail Page, page 7-9
- Answer Manager, page 7-10
- Define Answer Choice Page, page 7-10

Script Manager

The Script Manager displays the wizard scripts.

From this page, you can perform the following operations:

- Start to create a new wizard script, page 7-14
- Select a wizard script and then perform the following operations on it:
  - Edit, page 7-14
  - Copy, page 7-14
  - Delete, page 7-14
  - Graph, page 7-15
  - Deploy, page 7-15

Define Script Properties Page

In the Define Script Properties page, you define the following:

- Script Name
- Description (optional)
- Language

These fields represent the global script properties of the wizard script. For more details about these fields, see Defining Wizard Script Properties, page 7-15.

Panel Manager

The Panel Manager displays the panels in a wizard script.

From this page:

- You can create a new panel.
- You can rearrange the sequence of the panels.

- To edit, copy, or delete existing panels, select the panel and click the corresponding button.

Define Panel Information Page

In the Define Panel Information page, the main fields to define are the following:

- Panel Name

  This is the panel identifier.

- Panel Text

  This is the text that is either displayed or to be spoken when the script is run.

  If no text is required, delete the words "Enter text here."

  > **Note:** In some cases, question labels provide enough information and no panel text is required.

- Panel Text Style

  If you want to highlight the text in the Panel Text field as text that an agent should speak or as primary information to relay in the panel, select Spoken Text.

  If you want to indicate that the text in the Panel Text field provides instructions to the script end user, select Instructions.

  If you do not wish to apply any specific formatting, select Default.

- Text Alignment

  This is the default formatting for all paragraphs of the panel text.

- Question Alignment

  This is the default formatting for all questions included in the panel.

- Exit Panel Sequence
  - Go to the next panel in sequence
  - Go to a specific panel

    If you choose this option, then you must select the specific panel in the page that appears after you exit the Define Panel Information page, namely the Set Destination Panel page, page 7-8.

  - End Script

    This directs the script to terminate at runtime after the script user exits the

current panel.

When you exit the Define Panel Information page, the next page that appears depends on what you selected for the Exit Panel Sequence:

- If you selected "Go to a specific panel", the next page is the Set Destination Panel page, page 7-8.
- Otherwise, the next page is the Question Manager, page 7-8.

Set Destination Panel Page

The Set Destination Panel page enables you to direct the flow of a wizard script to a specific panel. It appears when you select the "specific panel" option in either of the following Wizard pages:

- Define Panel Information page, page 7-7

  In this case, use the Set Destination Panel page to name the next panel to appear when the script is run.

- Define Answer Choice page, page 7-10

  In this case, use the Set Destination Panel page to name the panel to appear when the specific answer is chosen.

In both cases, you can either select an existing panel or dynamically create a new, "placeholder" panel.

Select the appropriate radio button, then either select from the Select Existing Panel dropdown list, or enter a value in the New Placeholder Name field.

When you exit the Set Destination Panel page, if you chose the placeholder panel option, the placeholder panel is dynamically created.

If you designate a placeholder, you should later return to the Panel Manager to modify the placeholder panel's properties appropriately. Otherwise, the panel will contain the name you designate, but will contain only default panel properties provided by the Script Wizard (including panel text that reads "Enter text here"). For more information, see Placeholder Panels, page 7-24.

Question Manager

The Question Manager displays the questions in a script panel.

From this page:

- You can create a new question in the panel.
- You can rearrange the sequence of the questions in the panel at runtime.

- To edit, copy, or delete existing questions, select the question and click the corresponding button.
- You can go to the Panel Manager, page 7-6.

Define Question Main Properties Page

In the Define Question Main Properties page, you define the following:

- Question Name

  This is the question identifier that is stored in the database.

- Question Label

  The question label appears when the script is run and also identifies questions on reports.

- Type

  There are several options for question type, that fit into one of two broad categories:

  - Questions where script users enter text for the answer, such as a user name or a password.

    The options in this category are Text, Text Area, and Password.

  - Questions where script users select their answer or answers from a list of options.

    The options in this category are Radio Button, Dropdown, Checkbox Group, and Multi-Select List.

When you exit the Define Question Main Properties page, your choice of question type influences which page appears next in the Wizard:

- If you selected a question type with text-entry answers, the next page is the Define Question Detail page, page 7-9.
- Otherwise, the next page is the Answer Manager, page 7-10.

Define Question Detail Page

The Define Question Detail page appears only for questions that require their answers to be entered as text.

You must specify whether the answer is mandatory or optional.

You can also provide the following optional extras for the answer value:

- Default Value

When the script is run, the supplied default value appears, but script users can override the default.

- Validation

  The validation options are:

  - Integer Only

    The answer must be an integer.

  - Integer Range

    If you select this, you must also provide the low and high values in the Integer Range fields.

  - Date - Valid Date

    The date can be in the past.

  - Date - Not in the Past

When you exit the Define Question Detail page, you return to the Question Manager, page 7-8.

Answer Manager

The Answer Manager displays the answers for a question where script users must select one or more answers from a list of options.

From this page:

- You can create a new answer for the question.
- You can rearrange the display sequence of the answers when the question appears at runtime.
- To edit, copy, or delete existing answers, select the answer and click the corresponding button.
- You can go to the Question Manager, page 7-8.

Define Answer Choice Page

The Define Answer Choice appears only for questions where script users must select their answer or answers from a list of options.

In the Define Answer Choice page, you must provide the following details:

- Answer Value

This is the answer identifier which is stored in the database.

- Answer Label

  This is the text that is displayed when the script is run.

The other options available depend on whether the question type requires or allows single or multiple answers to be selected at runtime.

Radio buttons and dropdown lists require users to select a single answer; checkbox groups and multi-select lists allow for none, one, or more of the answer choices to be selected.

For radio button and dropdown question types, you have the following extra answer choice options:

- Whether the answer is to be the default answer for the question
- Which panel to branch to when the user selects the answer choice

  Note: Branching at the answer choice level overrides branching instructions provided at the panel level.

  The branching options are:

  - Go to next default panel
  - Go to a specific panel

    If you choose this option, then you must select the specific panel in the page that appears after you exit the Define Answer Choice page, namely the Set Destination Panel page, page 7-8.

  - End Script

    This directs the script to terminate at runtime after the script user selects the current answer choice.

For checkbox group and multi-select list question types, you have the extra option to specify whether your answer choice is to appear as a default value, that is, pre-selected at runtime. With these question types, you can have more than one default value. Script users can override default values and select further values at runtime.

When you exit the Define Answer Choice page, the next page that appears is determined as follows:

- If you selected "Go to a specific panel" for your answer choice, the next page is Set Destination Panel page, page 7-8.
- Otherwise the next page is the Answer Manager, page 7-10.

Accessing the Script Wizard

The Script Wizard is a feature of the Script Author Java applet. As such, you must first access Script Author in order to launch the Script Wizard to create, edit, or view wizard scripts.

> **Note:** You can view a list of wizard scripts from the Administration tab of the Scripting Administration console.

All remaining procedures described in this section assume that you have access to the Scripting Administration console and the Script Author Java applet, as described immediately below.

Use this procedure to open Script Author, from which you can access the Script Wizard.

Steps

1. Using an Oracle Applications 12 certified Web browser, navigate to the appropriate applications login URL for CRM Applications.

2. Log into Oracle Applications as a user with the Scripting Administrator responsibility.

   Optionally, if Scripting Administrator is not the default responsibility for this user, navigate to the profiles page and select Scripting Administrator as the current responsibility.

   The Scripting Administration console appears.

3. From the Home tab, click Launch Script Author.

   Oracle JInitiator launches in a separate window. Subsequently, Script Author opens as a Java applet in a separate window.

4. From the Script Author menu or toolbar, you can access the Script Wizard, as follows:

   1. **To create a wizard script from the menu:**

      File > New

      Tools > Wizard > Create Wizard Script

   2. **To create a wizard script from the toolbar:**

      New Script icon > Wizard Script

      Script Wizard icon > Create Wizard Script > Next

      All the options in cases 4.1 and 4.2 get you to the Define Script Properties page.

3. **To edit a wizard script from the menu:**

   File > Open

   Tools > Wizard > Edit Wizard Script

4. **To edit a wizard script from the toolbar:**

   Open a Script icon > Wizard Script

   Script Wizard icon > Edit Wizard Script > Next

   All the options in cases 4.3 and 4.4 get you to the Script Manager page, where you must select a script to continue processing.

Managing Scripts in the Script Wizard

The Script Manager page of the Script Wizard allows you to manage wizard scripts within the Wizard interface. From this page you can create new scripts, and edit, copy, delete, graph, and deploy existing wizard scripts.

For general information about the sequence of operations in the Wizard, see Overview of Wizard Script Pages and Operation Flow, page 7-3.

First-time users can also examine the Example of Starting to Create a Wizard Script, page A-1.

For details of the wizard pages dealing with scripts, select one of the following:

- Script Manager, page 7-6
- Define Script Properties Page, page 7-6

This section include additional information about the following script operations:

- Creating Wizard Scripts, page 7-14
- Editing Wizard Scripts, page 7-14
- Copying Wizard Scripts, page 7-14
- Deleting Wizard Scripts, page 7-14
- Graphing Wizard Scripts, page 7-15
- Deploying Wizard Scripts, page 7-15
- Defining Wizard Script Properties, page 7-15

See Also

- Saving and Deploying Wizard Scripts, page 7-20

Creating Wizard Scripts

1. To create a wizard script and its component objects, follow the steps described in Overview of Standard Wizard Script Setup, page 7-5.

   Note: When you create a wizard script, the first page that appears is the Define Script Properties page, not the Script Manager.

2. You can only save a wizard script if it is syntactically correct.

Editing Wizard Scripts

1. You can view and edit an existing wizard script at any time. If a previously deployed wizard script is being used as the questionnaire script for an active survey campaign, it is locked. In this case, while the Wizard will allow you to modify and save the script, you will not be able to deploy it until the script is unlocked or you change the name of the script.

2. If you make changes that put defined objects at risk - for example, when changing a question's type from a radio button, which requires answer choices, to a text area, which does not - you will be advised accordingly.

3. As when creating wizard scripts, you can only save the script when it is syntactically correct.

4. You cannot open graphical scripts from the Wizard.

Copying Wizard Scripts

1. When you copy a wizard script, the initial name of the copied script is the original script name preceded by the words "Copy of."

Deleting Wizard Scripts

1. Scripts must be unlocked in order to be deleted, that is, they must not be part of an active survey campaign.

   For information on removing a script from the database using the Scripting Information console, refer to the Oracle Scripting Implementation Guide.

Graphing Wizard Scripts

1. You can graph a script either explicitly from the Script Manager, or as part of the process of saving a wizard script.

   In the latter case, when you choose either "Save and exit" or "Save, deploy, and exit", you are given the choice to create a graphical script copy. For details, see Saving and Deploying Wizard Scripts, page 7-20.

2. When you graph a wizard script, the following occurs:

   - The original wizard script remains, with its original name.
   - A separate, graphical script is created whose initial name is the wizard script name preceded by the words "Copy of."

Deploying Wizard Scripts

1. Deploying a script makes it available to execute in the Scripting Engine. If you set up and activate a survey campaign, you will subsequently also be able to execute the script in a Web browser using the Scripting Engine Web interface.

2. If a wizard script with the same name has already been deployed, then deploying a different version replaces the previous script, making it inactive. The previous version will not be available to execute.

   As well as deploying a wizard script from the Script Manager, you can also deploy during the process of saving. For more information, see Saving and Deploying Wizard Scripts, page 7-20.

Defining Wizard Script Properties

Each wizard script has three configurable global properties, which appear - and which you can edit - in the corresponding fields of the Define Script Properties page:

- The Script Name field contains the name by which a script is identified in the applications database.

  You can use dash, space, and underscore characters in the name, but avoid the use of slash (/) or backslash (\), colon (:), percent (%), asterisk (*), question mark (?), explanation point (!), tilde (~), or left or right angle bracket (< or >) characters.

- The Description field is optional, and for information only.

- The Language field reads and displays available languages from the Oracle RDBMS FND_LANGUAGES table. Select the appropriate language for your environment, or leave the default AMERICAN setting.

These properties equate to the global script properties accessed in a graphical script

from the Script Author. A graphical script includes three more configurable options as global script properties: boolean controls which control footprinting, answer collection, and script suspendability. For wizard scripts, these boolean options are set to true by default for each wizard script. To modify any of these properties, you must graph the script and disable the appropriate option.

Managing Panels in the Script Wizard

The Panel Manager page of the Script Wizard allows you to view and manage existing panels in your wizard script. From this page you can reorder existing panels in your wizard script, as well as create, edit, copy, and delete panels.

For general information about how panel operations fit in the creation of a wizard script, see Overview of Standard Wizard Script Setup, page 7-5.

For general information about the sequence of operations in the Wizard, see Overview of Wizard Script Pages and Operation Flow, page 7-3.

First-time users can also examine the Example of Starting to Create a Wizard Script, page A-1.

For details of the wizard pages dealing with panels, select one of the following:

- Panel Manager, page 7-6
- Define Panel Information Page, page 7-7
- Set Destination Panel Page, page 7-8

General Notes

1. Wizard scripts do not require every panel to contain a question. A Continue button is generated on a panel automatically, whether there are questions in the panel or not.

Reordering Panels

1. When two or more panels are defined in a wizard script, you can reorder panel sequence.
2. For panels for which the exit panel sequence is the next panel in sequence, reordering the panel sequence changes the flow of the script at runtime. For example, moving a panel up in the Panel Manager makes it display sooner at runtime.
3. If the exit panel sequence specifies a designated panel or the end of the script, reordering panel sequence will have no effect at runtime.

Deleting Panels

1. Deleting a panel permanently removes the panel from the script, and automatically reroutes the flow of the script to the next panel in sequence.

2. Deleting a panel also removes any components associated with a panel, including panel questions, answer choices for a question, and branching associated with specific answer choices.

3. Each wizard script must contain at least one panel. If you delete the only panel in a wizard script, you must create a new panel before the script can be saved or deployed.

Managing Questions in the Script Wizard

The Question Manager page of the Script Wizard allows you to view and manage panel questions in your wizard script. From this page you can reorder existing questions within a selected panel in your wizard script, as well as create, edit, copy, and delete questions.

For general information about how question operations fit in the creation of a wizard script, see Overview of Standard Wizard Script Setup, page 7-5.

For general information about the sequence of operations in the Wizard, see Overview of Wizard Script Pages and Operation Flow, page 7-3.

First-time users can also examine the Example of Starting to Create a Wizard Script, page A-1.

For details of the wizard pages dealing with questions, select one of the following:

- Question Manager, page 7-8
- Define Question Main Properties Page, page 7-9
- Define Question Detail Page, page 7-9

General Notes

1. A panel can have any number of questions.

2. Panels containing only panel text do not require questions to be defined.

3. For questions that require text entry answers, you must define question details; for all other types of question, you must define one or more answer choices.

Editing Questions

1. If you modify the question type from one that contains answer choices (radio button, dropdown list, checkbox group, or multi-select list) to one that does not (text field, text area, or password field), any defined answer choices are permanently removed from the data dictionary.

2. If you modify the question type from one that allows branching in the script based on answer choices (radio button or dropdown list) to any other question type, any defined answer choice-level branching for this panel is permanently removed. The script will then follow branching instructions at the panel level, and for any other questions defined in this panel.

Copying Questions

1. When you copy a question, be aware that distinct branching of the script based on an answer choice can only be included in one question in each panel.

   If the question you want to copy includes branching associated with an answer choice, branching properties will only be included in either the original question, or the copy. You decide the matter with your response to the "Only 1 Question Can Determine Branching for this Panel" page.

Deleting Questions

1. Deleting a question permanently removes any components associated with the question, including answer choices for the question, and branching associated with specific answer choices.

2. Deleting a question with answer choices that control branching to distinct panels will affect the flow of the script.

   - Specific business rules incorporated into the script will be removed, and the destination or progression of the script when exiting that panel will only be controlled by the exit panel sequence designated for that panel.

   - If a question you delete contains an answer choice that specifies flow to a specific panel, and that panel is not the destination of any other panels in the script, the panel becomes an *orphan*, that is, the panel is no longer displayed or evaluated for any session of the script at runtime.

Managing Answer Choices in the Script Wizard

The Answer Manager page of the Script Wizard allows you to view and manage answer choices (sometimes referred to as lookup values) for a specific question in a specific panel in your wizard script.

Question types in a wizard script that display multiple answer choices include radio buttons, dropdown lists, checkbox groups, and multi-select lists. For these question types, the Script Wizard routes you to the Answer Manager page automatically when creating a question. From this page you can reorder existing answer choices within a selected question, as well as create, edit, copy, and delete answer choices

For general information about how answer operations fit in the creation of a wizard script, see Overview of Standard Wizard Script Setup, page 7-5.

For general information about the sequence of operations in the Wizard, see Overview of Wizard Script Pages and Operation Flow, page 7-3.

First-time users can also examine the Example of Starting to Create a Wizard Script, page A-1.

For details of the wizard pages dealing with answers and answer choices, select one of the following:

- Answer Manager, page 7-10
- Define Answer Choice Page, page 7-10
- Set Destination Panel Page, page 7-8

General Notes

1. Each answer choice in a question may contain separate distinct branching instructions to control flow in the script.

2. Only one question in a panel can determine branching based on instructions provided at the answer choice level.

   If you designate one question (for example, Question 1) in a panel that uses this functionality, and then you attempt to set a destination panel for an answer choice in a different question (for example, Question 2), you will be prompted to tell the Wizard whether to override the branching instructions in the first question with the branching rules you want to define for Question 2.

3. In the Wizard, you cannot populate answer choices with values from a table (table lookups), from a previous database query (cursor lookups), or as a result of a command (command lookups). The wizard only allows you to specify predefined, hard-coded answer choices.

   To populate questions with dynamic answer choices (table, query, or command lookups), you must graph the script, and, using Script Author graphical tools, access the Lookups tab of the data dictionary for the specific question.

   After you graph a wizard script and modify the graphical script with graphical tools, you cannot view, edit, or deploy the modified script from the Script Wizard.

Editing Answer Choices

1. If you modify the branching properties associated with an answer choice, the flow of the script at runtime is modified accordingly.

Deleting Answer Choices

1. Deleting an answer choice permanently removes the answer choice from the question.

   Business rules incorporated into the script based on the presence of the answer choice (specifically, branching based on selection of that answer choice at runtime) are also removed. This can change the destination or progression of the script.

   In lieu of specific rules based on selecting the answer choice, the script will rely upon the exit panel sequence designated for that panel, or branching associated with other answer choices in the question.

Saving and Deploying Wizard Scripts

You can save a wizard script from any page in which the Save button is enabled. Wizard scripts can only be saved when they are syntactically correct and contain at least one panel.

When you select any of the available options in the Script Wizard to save a script, you *publish* the script. In other words, you cause the information currently entered in the Script Wizard for that script to be written to the IES_DEV_SCRIPTS table of the applications database. You cannot save a wizard script in the file system of a local or network drive, as you can with a graphical script.

After a wizard script is saved, you can open it from the Script Wizard, edit it, copy it, delete it, create a graphical script copy, or deploy it.

Scripts cannot be executed until they are deployed from Script Author. From the Script Wizard, scripts can be deployed using the "Save, Deploy, and Exit" command.

This section includes the following topics:

Understanding Save Options

When a wizard script contains at least one panel and is syntactically correct, the Save

button is enabled, allowing you to execute one of the save options.

You can save a wizard script from any page in which the Save button is enabled.

The Save options for a wizard script are as follows:

- Save and Continue Editing
- Save and Exit
- Save, Deploy and Exit

General Notes

1. **Each time that you save a wizard script, you can save it with a different name.**
2. Upon selecting Save, if the script contains placeholder panels, the Placeholder Panels Have Not Been Edited page appears. This page warns you that, during the creation of the wizard script, a panel was been included in the flow of the script that contains a default text message (Enter text here). No questions or predefined answer choices are included in placeholder panels.

   For production scripts, replace the contents of each placeholder panel as appropriate by selecting Back, returning to the Panel Manager, and editing any placeholder panels. To continue after receiving the placeholder panel warning, click Next.

   Upon selecting Save (or upon leaving the placeholder panel warning page, if appropriate), the Save Script <script name> page appears.

   Selecting Yes in this page results in the Save Script <script name> page, in which the words "Copy of" are prepended to the existing script name.
3. The ability to graph a copy of the wizard script is available to the "Save and Exit" and "Save, Deploy and Exit" options.

Save and Continue Editing

Selecting "Save and Continue Editing"saves all the changes you have made to the current wizard script, publishes the script in the applications database, and routes you to the Panel Manager page, from which you can make additional changes.

Save and Exit

Selecting "Save and Exit"does the following:

- Saves all the changes you have made to the current wizard script.
- Publishes the script in the applications database.

- Routes you to the Create Graphical Script Copy page, from which you can elect to graph the script to view or modify it using Script Author graphical tools.
- Confirms the script save operation.
- Exits the Script Wizard.
- Displays the Script Author visual layout:
  - If you graphed the script, the copy of the wizard script is displayed. The script view shows the entire graph in the page.
  - If you did not graph the script, displays Script Author application with no script open.

Save, Deploy and Exit

Selecting "Save, Deploy and Exit" does the following:

- Saves all the changes you have made to the current wizard script.
- Publishes the script in the applications database.
- Routes you to the Create Graphical Script Copy page, from which you can elect to graph the script to view or modify it using Script Author graphical tools.
- Confirms the script save operation.
- Exits the Script Wizard.
- Displays the Script Author visual layout:
  - If you graphed the script, the copy of the wizard script is displayed. The script view shows the entire graph in the page.
  - If you did not graph the script, displays Script Author application with no script open.
- Attempts to deploy the script to the applications database, and displays a message in the Script Author status area regarding the success or failure of the deployment.

Determining Flow with the Script Wizard

Script flow is controlled at two different levels for wizard scripts: panel level and answer choice level.

At the panel level, you can determine flow of the script based on three conditions,

specified in the panel's exit panel sequence.

- You can direct the script to the next panel in the panel sequence. This is the default behavior of the application. This uses a default branch to progress the script at runtime to the next sequential panel (as viewed and controlled in the Panel Manager page).

  From the Define Panel Information panel, for the Exit Panel Sequence field, select **Go to the next panel in sequence**.

- You can direct the flow of the script to a designated panel. This uses a distinct branch from the source panel to the subsequently identified panel. You can designated an existing (already defined) panel, or you can create a placeholder panel. The placeholder panel, a Script Wizard-specific innovation, has certain caveats associated with it.

  From the Define Panel Information panel, for the Exit Panel Sequence field, select **Go to a specific panel**.

- You can end the script. This uses a default branch to a termination node on the root graph.

At the answer choice level, you have the same three choices, in the Define Answer Choice page. Again, the default behavior of the application is to route the script to the next sequential panel. Answer choice-specific script flow instructions are invoked by the Scripting Engine only when the specific answer choice containing those instructions is selected at runtime by the script end user. These options override the panel-level instructions.

Wizard Script Flow Strategies

This section includes the following topics:

Distinct Branching and Wizard Scripts

When using distinct branching in wizard scripts, you can use one of the following approaches:

1. Determine the flow of the entire script, create all panels, and then modify the panel properties to branch accordingly.
2. Build the script, including routing instructions to placeholder panels, and combine careful review of each panel, question, and answer choice option with thorough testing of each possible path in the script.

3. You can create the wizard script using either of the preceding approaches, graph it, and review the graphical script so that you can get a visual understanding of the script flow.

> **Note:** Remember that once you make changes to a graphed script, you cannot edit it using the Script Wizard. Thus, if it is your intention to continue development in the Script Wizard, you can determine from the graphical script which changes are required, and return to the wizard script to implement the necessary changes.

When using distinct branching, you must be particularly careful of the exit panel sequence and the answer choice branching options you select.

For example, you can create a script with the following objects:

- A panel called Ice Cream.
- A question called Flavor.
- Three answer choices: Chocolate, Vanilla, and Other.

You can branch all three answer choices to different new panels by creating placeholder panels (one for each flavor) in the Answer Manager.

If you create only two placeholder panels, Destination_for_Chocolate and Destination_for_Vanilla, and for the third flavor answer choice, Other, you specify "Go to next default panel," then selecting the third answer choice at runtime will take you to the next default panel.

If this is a new wizard script, then the default panel will be the second panel created - or your first placeholder panel. Thus, inadvertently, choosing Chocolate or Other will both result in the Destination_for_Chocolate panel.

Placeholder Panels

Placeholder panels are created in a wizard script when you designate a particular panel as the branching destination for a panel or answer choice. You can designate destination panels from two locations: as the exit panel sequence for a panel, or when defining an answer choice.

At the time of creation, placeholder panels are provided with a panel name and a branching destination only. The remaining panel properties include the standard default settings for a panel. These panel attributes are listed in the table below.

Panel Attribute	Description
Panel Text	Enter text here
Panel Text Style	Spoken Text
Text Alignment	Left
Question Alignment	Left
Exit Panel Sequence	Go to the next panel in sequence

You must return to the placeholder panel from the Panel Manager page to modify any of these attributes. At the very least, you should delete the default panel text "Enter text here."

Unique Placeholder Panel Attributes

Placeholder panels are the only panels listed in the Panel Manager page that contain a value of Yes in the Place Holder column.

If a placeholder panel is the last panel created in a script, its destination is Exit in the Panel Manager page, indicating that the script will terminate after leaving that panel. If this is not your intention, you must modify the panel's properties, or add another panel. Adding another panel will result in the new panel being the destination of the placeholder panel, unless you change the order of the placeholder panel in the Panel Manager page.

What Do Wizard Scripts Contain or Exclude?

While wizard scripts are constructed by responding to prompts in the Script Wizard, the script is technically constructed of the same elements as a graphical script. The Script Wizard ensures that all scripts created and saved are syntactically correct.

What Do All Wizard Scripts Contain?

The properties that you will find in all wizard scripts are detailed below:

- Wizard scripts always contain at least one panel.
- Each panel includes an automatically generated Continue button.
- Wizard scripts always contain only a single group (Disconnect).

  By default, all wizard scripts include the Disconnect group. This group represents a subgraph with no panels (a start node and termination node, connected with

default branching).

This group contains the WrapUpShortcut name in the shortcut group property, which enables the Disconnect button in the Scripting Engine agent interface at runtime.

- Wizard scripts always contain two start nodes (one on the root graph, and one in the Disconnect group).
- Wizard scripts always contain two termination nodes (one on the root graph, and one in the Disconnect group).
- Wizard scripts always contain default branching.
- Wizard scripts include all boolean global script properties (footprinting, answer collection, and suspendability) enabled by default.

  Unless subsequently graphed and further edited as a graphical script, scripts created with the Script Wizard cannot disable any of the boolean properties described earlier.

What May Wizard Scripts Include?

- Wizard scripts may include explicitly defined questions.

  When defining a panel, a Continue button is automatically generated for each panel, whether or not the panel contains any questions.

- Wizard scripts may include distinct branching.

  You can designate a specific panel as the next panel to branch to when you exit a panel at runtime.

  You can also designate a specific panel to branch to when you specify a particular answer to a question at runtime.

- Wizard scripts may include constant commands to provide default answer choices at runtime.

  Providing an entry in the Default Value field of the Define Question Detail page of a wizard script will automatically provide a constant command for a panel question at runtime.

- Wizard scripts may include references to best practice Java methods in the NodeValidation class by selecting options in the Define Question Detail wizard page. These include response validation for an integer or date provided as an answer at runtime (validation that the answer provided is an integer only, or is an integer within a provided range, or is a valid date, or is a valid date later than the current SYSDATE), as well as validation that an answer is provided at runtime (for example, a question is not left as a null value by the user).

What Do All Wizard Scripts Exclude?

- Wizard scripts do not contain blocks.
- Wizard scripts do not contain groups other than the Disconnect group.
- Wizard scripts do not include panels in the Disconnect group.
- Wizard scripts do not use conditional or indeterminate branching.
- Wizard scripts do not include custom Java or PL/SQL code.
- Wizard scripts do not include global script pre-actions or post-actions.
- Wizard scripts do not contain the single checkbox nor button user interface controls for questions in a panel.

Unless subsequently graphed and further edited as a graphical script, scripts created with the Script Wizard cannot include any of the features described earlier.

8

Using Graphical Scripts

This chapter covers the following topics:

- Introduction
- Graphical Script Layout
- Overview of Objects and Branches
- Script Author Terminology
- General Canvas Operations
- Working with Graphical Scripts
- Working with Panels
- Understanding Answer Properties and Answer Types
- Working with Answers and Answer Properties
- Working with the Panel Layout Editor
- Working with Groups
- Working with Blocks
- Using Subgraph Information in Blocks and Groups
- Working with Branches
- Working with Database Connections
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying Scripts
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

Introduction

In the Script Author, you can create and modify graphical scripts. This chapter provides information on the following graphical script features:

- The layout of the screen areas in the Script Author when you work on graphical scripts
- The graphical script elements and their properties
- The operations to create and maintain graphical scripts

Graphical Script Layout

The main components of the Script Author graphical script layout are the following:

- Three toolbars
- The main canvas which displays the visual objects and the branches that represent the script flow

The content of the left hand side of the window is controlled by two tabs, Project and Syntax, that appear at the foot of the window.

With the Project tab selected, the left hand side of the window displays two panes, both untitled:

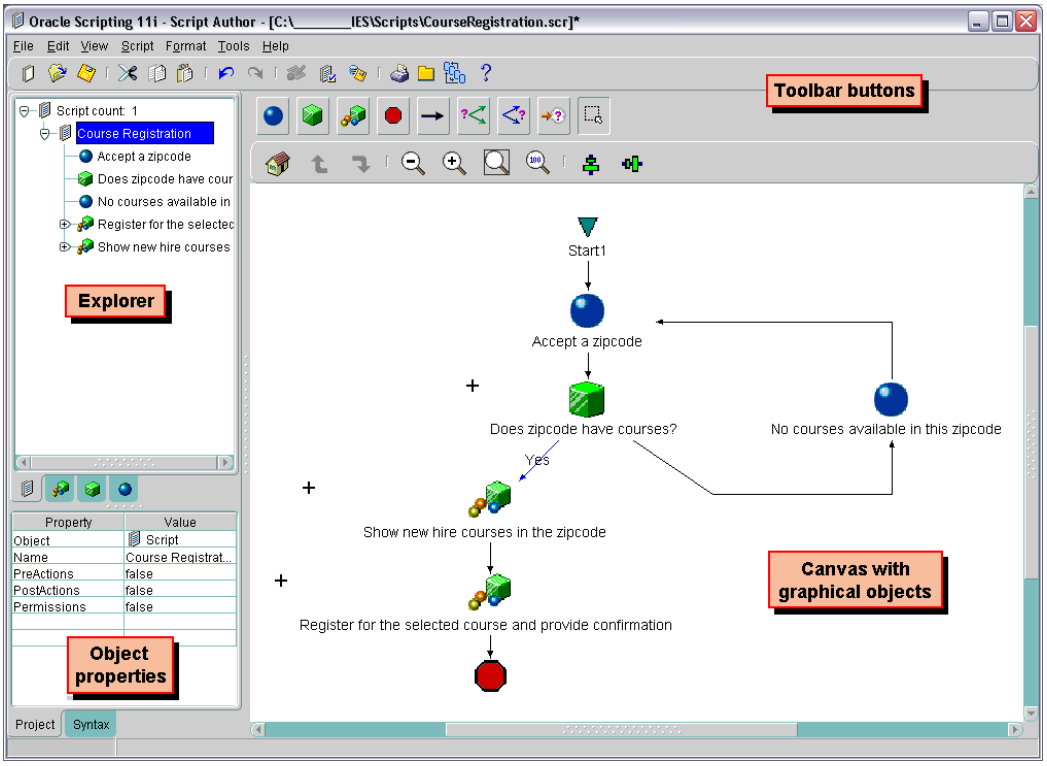
- The Explorer pane, that displays an expandable list of the script objects
- The Object Properties pane, that displays the properties of an object highlighted either in the canvas or in the Explorer pane

The Syntax tab appears automatically when you perform a syntax check, and displays syntax errors.

Clicking on any listed error will cause a section of the canvas to be replaced with the Debug pane. The debug pane lists any available information about the selected problem.

At the very bottom of the Script Author work area, below the canvas and the Project and Syntax tabs, is a status bar. This indicates the status after a defined action such as checking syntax, saving a script, or deploying a script to the database.

The screenshot that follows shows a graphical script layout with the three toolbars, Explorer pane, Object Properties pane, and a graphical script in the canvas. The graphical script contains two panels, a block whose main purpose is to direct script flow in two different directions, two groups that represent two extra sets of processing contained in their subgraphs (details not visible in the top graph), and a termination node.



Graphical Script Toolbars

The three toolbars used for graphical scripts are the following

- Top Toolbar, page 8-3
- Object and Branch Toolbar, page 8-4
- Navigation and Alignment Toolbar, page 8-6

Top Toolbar

The top toolbar enables general scripting functions, and also contains icons for standard features, such as cut and paste.

The top toolbar buttons fall into three categories:

- Script Buttons, page 8-3
- General Buttons, page 8-4
- Miscellaneous Buttons, page 8-4

Script Buttons

The script buttons are the standard buttons that enable you to create a new script, open

an existing script, and save the current script.

General Buttons

The general buttons of the top toolbar enable you to perform the standard operations of cut, copy, paste, undo, and redo on visual script objects in the canvas. There is no limit to the number of operations that you can reverse or redo.

Miscellaneous Buttons

The following image shows the miscellaneous buttons of the top toolbar.

The table that follows describes the function of each of these buttons.

Button	Function
Panel Layout	Opens up the panel layout window for a highlighted panel. Only enabled when a panel is highlighted.
Check Syntax	Checks the script syntax. The syntax errors will appear in the Syntax pane.
Deploy Script to Database	Deploys the script to the database.
Print	Prints the current canvas graph.
Command Library	Displays commands that can be added to the script.
Script Wizard	Launches the Script Wizard part of the Script Author.
About Script Author	Displays level information.

Object and Branch Toolbar

The Object and Branch toolbar consists of the following:

- Canvas Object Buttons, page 8-5

- Branch and Toggle Select Mode Buttons, page 8-5

Canvas Object Buttons

From the Object and Branch toolbar, you create the main visual script objects using the following buttons:

- Panel Insertion Mode
- Block Insertion Mode
- Group Insertion Mode
- Termination Point Insertion Mode

Click the appropriate button, then click in the canvas: the corresponding object appears.

> **Note:** If you have sticky mode set - see Branch and Toggle Select Mode buttons - you can click one of these canvas object buttons, then click several times in the canvas to create multiple objects of the selected type.

> **Note:** For more details of sticky mode, see Enabling and Disabling Sticky Mode, page 8-20.

Branch and Toggle Select Mode Buttons

To the right of the canvas object buttons on the Object and Branch toolbar are four branch buttons and a single Toggle Select Mode button.

- Default Branch
- Distinct Branch
- Conditional Branch
- Indeterminant Branch
- Toggle Select Mode

For general information about branches, see Branches, page 2-8.

Toggle Select Mode

The Toggle Select Mode button is used mainly in conjunction with sticky mode processing to indicate whether any canvas object button has been selected, and to allow for general canvas object editing. For more details, seeEnabling and Disabling Sticky Mode, page 8-20.

Navigation and Alignment Toolbar

The Navigation and Alignment toolbar consists of the following sets of buttons:

- Inter-Graph Buttons, page 8-6
- Zooming Buttons, page 8-6
- Aligning Buttons, page 8-6

Inter-Graph Buttons

Groups and blocks implicitly contain child graphs. The inter-graph buttons control movement between graphs:

- Go to root graph
- Go to parent graph
- Go to child graph

Zooming Buttons

The Navigation and Alignment toolbar has are four buttons that control the zooming of the current graph, and fitting the graph into the canvas.

- Zoom Out
- Zoom In
- Zoom to Fit Graph in Window
- Zoom to 100%

Aligning Buttons

You can arrange visual script objects by first selecting them, and then clicking the appropriate aligning button to align them either vertically or horizontally.

- Align vertically
- Align horizontally

Overview of Objects and Branches

The main objects in a graphical script are:

- Panels, page 8-7
- Blocks, page 8-7
- Groups, page 8-7

  These are configurable objects, in that their properties are viewable and editable.

  The following objects are non-configurable.

- Start nodes, page 8-7
- Termination nodes, page 8-7

Panels, groups, and blocks are processing units that tell the script what to do. These objects must be connected with branches, page 8-8 to direct the flow of the script at runtime.

A graphical script can consist of one or more graphs. Each Script Author graph must have one start node, with a default branch leading to the first script object, and at least one termination node.

Panels

Panels display information (such as text, an image, a hyperlink, a variable, or question UI controls) to the script user at runtime, and provide a method to accept information (answers) from the script end user.

Blocks

Blocks are containers that enable you to connect to a database during script execution, (the script can query, update, or insert information in database tables), or to execute an API command.

Groups

Groups are used to logically group a section of the script's functionality, access a section of the script in runtime using a shortcut button, or for functionality that is the target of a Java method in runtime associated with an indeterminate branch.

Start Nodes

Whenever you create a graph, it automatically contains a start node. Start nodes cannot be explicitly created, nor created, nor do they contain viewable properties. They must be attached using a default branch to the first executable object in the graph.

Termination Nodes

Termination nodes are objects which indicate the end of processing on a script graph.

They are non-configurable, with no viewable properties. Except for graphs that contain an indeterminate branch, every graph must have at least one termination node; each termination node must be attached with a branch to another canvas object.

Branches

A branch is used to connect objects. The type of branch used determines the next object in the script flow.

There are four branch types:

- **Default** - unconditional branch which directs flow to the designated object
- **Distinct** - branch where destination depends on answer provided to a question
- **Conditional** - branch whose destination is reached if a Boolean expression returns a value of true
- **Indeterminate** - branch whose destination is evaluated at runtime

> **Note:** The terms *indeterminant* and *indeterminate* are used interchangeably in Oracle Scripting.

> **Note:** Two branch types (default and distinct branches) contain all of the information required to direct the flow of the script. The other two branch types (conditional and indeterminate branches) require the script developer to associate a Script Author command with the branch; the return value of the condition or expression for that command (determined at runtime when the branch is reached in the flow of the script) dynamically determines the flow of the script.

For general information about branches and the branch types, see Branches, page 2-8.

Script Author Terminology

The term *blob* is also used in Script Author as a generic term for panels, groups, and blocks. In this chapter the terms "*blob*"and "*configurable object*" are used interchangeably.

> **Note:** Branches are generally not considered as configurable objects, even though some branch properties can be edited.

The term *graphical script object* is used for any object that appears on a script canvas, that is, any one of the following:

- A configurable object
- A branch
- A start node
- A termination node

This chapter concentrates on how to use the Script Author to define the attributes and properties of the configurable script objects and branches. More general information about the objects themselves appears in the Understanding Script Author chapter.

General Canvas Operations

As with many visual tools, there are several standard graphical script operations you can perform in the Script Author.

The toolbars enable a variety of functions and operations, some following industry-standard conventions, some more specific to Oracle Scripting objects and processes.

This section describes these operations in general, as a set of notes for each operation or group of operations. These notes contain links to more specific descriptions in other sections as appropriate.

This section consists of the following topics:

Inserting Objects and Branches

The main script objects that you can insert fall into two categories:

- The main visual script objects: Panel, Group, Block, Termination Node.
- Branches

The basic process of inserting a *main visual script object* is:

1. Select the object type on the toolbar.

2. Click on the canvas:

The basic process of inserting a *branch* is:

1. Select the branch type on the toolbar.
2. Click the canvas object where the branch begins.
3. For a default, distinct, and conditional branch, continue dragging to the destination object; for an indeterminate branch, continue dragging into empty space on the canvas.

> **Note:** Oracle Scripting has a *sticky mode* feature, that allows for an optimized way of creating several objects and branches of the same type. For details of sticky mode, and how this affects canvas object operations, see Enabling and Disabling Sticky Mode, page 8-20.

See Also

- Enabling Automatic Popup of Properties Windows at Object Creation Time, page 8-10

Enabling Automatic Popup of Properties Windows at Object Creation Time

To optimize the process of adding properties to newly-inserted configurable objects (panels, groups, and blocks) and branches, you can select one or both of the following menu options to allow for the automatic popup of the appropriate Properties window when the canvas object is inserted:

- View > Select the checkbox "Popup on Blob Creation"
- View > Select the checkbox "Popup on Branch Creation"

Editing Objects and Branches

There are two kinds of editing for all blobs and branches:

- Industry-standard functions: Cut, Copy, and Paste
- Specifying the object and branch properties

> **Note:** The Oracle Scripting sticky mode feature has some considerations that affect the editing of objects and branches. For details, see Enabling and Disabling Sticky Mode, page 8-20.

Using Industry-Standard Functions

Cut, Paste, and Copy behave in the standard way: for example you can cut or copy an object, then paste it once or several times in the canvas.

The basic process for these functions is:

1. Right click a canvas object or branch.
2. Select the appropriate option: Edit > Cut, Copy, or Paste.

Specifying Object and Branch Properties

You add and edit properties in a Properties window specific to an object or branch type. You can ensure that the Properties window is automatically launched when the object or branch is created (see Enabling Automatic Popup of Properties Windows at Object Creation Time, page 8-10) or you can launch it manually at any time, as follows:

1. Right click the canvas object or branch.
2. Select either Edit Blob Properties or Edit Branch Properties as appropriate.

For more information, see the following topics:

- Panel Properties and Attributes, page 8-29
- Group Properties and Attributes, page 8-66
- Block Properties and Attributes, page 8-70
- Branch Properties, page 8-82

Deleting Objects and Branches

Apart from start nodes, you can delete any object or branch from a graph.

> **Note:** You can indirectly delete a start node in a subgraph of a block or group, by deleting the parent block or group. You cannot delete the start node of the root graph.

Deleting an object deletes all properties, edges (outgoing branches), and subgraphs associated with the object.

Deleting a branch deletes all properties associated with the branch.

> **Note:** The Oracle Scripting sticky mode feature has some considerations that affect the deleting of objects and branches. For details, see Enabling and Disabling Sticky Mode, page 8-20.

The basic deletion process is:

1. Right click a canvas object or branch.
2. Select Edit > Delete.

Standard User Interface Operations

The Script Author includes a number of common user interface operations that generally follow industry standards. Some are performed entirely within the canvas, others combine canvas operations with one or more toolbar buttons.

> **Note:** In this section, the term *object* refers to all blobs, nodes, and branches.

> **Note: Before you perform any of the standard user interface operations in this section, if the Toggle Select Mode button on the Object and Branch toolbar displays a raised image, you must click the Toggle Select Mode button.**

Selecting and Deselecting Objects

You can select or deselect one or more objects on a single canvas using standard user interface functions.

- Select an object by clicking it
- Select several objects by clicking and dragging a rectangle around them
- Select individual objects together by depressing the Ctrl or Shift key as you click each individual object
- Deselect an object by clicking away from it
- Deselect all selected objects by clicking in empty canvas space
- Deselect an object from a group of selected objects by depressing the Ctrl key as you click the object that you want to deselect

Moving Objects

You can move any objects that you have selected by dragging them. Any branches connected to objects move with the objects.

Aligning Objects

Select the objects that you want to organize, and click the appropriate Align button in the Navigation and Alignment Toolbar, page 8-6.

Zooming and Fitting Graph to Window

You can increase or decrease the magnification level of the canvas objects by using the Zoom buttons in the Navigation and Alignment Toolbar, page 8-6.

To display all the canvas objects for a graph, use the button Zoom to Fit Graph in Window.

Navigating Between the Graphs of a Script

All graphical scripts have a root graph. Groups and blocks on any graph have subgraphs. You can navigate between the graphs of a script by clicking the appropriate inter-graph button in the Navigation and Alignment Toolbar, page 8-6.

Working with Graphical Scripts

This section consists of the following topics:

- Creating New Graphical Scripts, page 8-13
- Opening Graphical Scripts, page 8-14
- Saving Graphical Scripts, page 8-15
- Importing Scripts, page 8-17
- Exporting Script Groups as Separate Script Files, page 8-18
- Printing Graphs, page 8-20
- Enabling and Disabling Sticky Mode, page 8-20
- Closing Scripts and Exiting Script Author, page 8-22
- Defining Global Script Attributes, page 8-22

Creating New Graphical Scripts

Note: This section concentrates on graphical scripts. For creating new wizard scripts, see Accessing the Script Wizard, page 7-12.

Steps:

1. Select either of the following:
   - Using menu options, File > New

- New Script toolbar icon

2. In the New Script window, select Graphical script.

   A new graphical script, with a dummy script name of Untitled1, and with one start node, appears in the Script Author canvas.

Opening Graphical Scripts

> **Note:** This section concentrates on graphical scripts. For opening and working with wizard scripts, see Accessing the Script Wizard, page 7-12.

From within Script Author, graphical scripts can be opened from either of the following:

- The file system, on any local volume
- The applications database, whether the scripts are published or deployed.

Within the database, there is a distinction between *published* scripts and *deployed* scripts. Published scripts are saved in the database, but not necessarily executable. Deployed scripts are also saved in the database, but can be executed.

Steps:

1. Select either of the following:
   - Using menu options, File > Open
   - Open Script toolbar icon
2. In the Open Script window, select Graphical script.
3. In the Script Chooser window, select either File System or Database
4. If you selected File System:
   - Select the script file from the file system, then click Open.

   If you selected Database:
   - Select Published or Deployed.
   - Click Refresh List.
   - Select the script from the database, then click Open.

If the script was published or deployed from an earlier version of Script Author, a Warning dialog appears, noting that if you save the file with the same name, you may not be able to open the script in older versions. Click OK to continue.

The designated script appears on the canvas.

Saving Graphical Scripts

From Script Author, you can save a graphical script to the file system of a local volume or mounted storage medium.

You can also publish a script to the applications database. Publishing a script simply means *storing* the script file in the database. A published script is not deployed, that is, it is not executable.

To determine whether a script needs to be saved, examine the title bar of the Script Author applet. If an asterisk (*) appears to the right of the script name, changes have been made to the script that have not yet been saved.

New scripts are automatically labeled Untitled in the title bar. These initially include only a start node. If no asterisk appears, no objects have been added or configured.

Script Name and File Name

All scripts have a script name - which is a script property - and this name identifies the script in the database, and is used when you choose to run the script.

If you save the script to the file system, you must also provide a *file name* for the script. This is distinct from the script name. For convenience you may make the file name and the script name the same.

You can save a script to a local or network file system using the current name and location (if the script has previously been saved), or you can save a copy of the script to a current or new location, using a different file name.

Note: If you wish to save the script with a different script name, ensure that you change the name of the just-opened script in the global script properties (File > Script Properties).

Note: If you want to save the script to a new file location, change the file name of the script as well (File > Save As).

Steps:

1. Select either of the following:
   - Using menu options, File > Save or File > Save As

- Save Script toolbar icon

2. In the Script Chooser window, select either File System or Database.

3. If you selected File System:

   - In the File Type field, select Oracle CRM script file (.script, .scr).

   - In the File Name field, type the name you want to assign to this script at the file system level.

     For ease of use, provide the file name with a file extension of .SCRIPT or .SCR.

     When opening a graphical script, Script Author automatically filters out files with no file extension, or with a different extension than these defaults.

     If you selected Database:

   - The global script name and language properties automatically populate in the Script Name and language fields. These cannot be changed.

4. Click Save.

   The script is saved either as a file or in the database.

Guidelines for Saving to the File System

- When you select File > Save for a previously saved script, the current version of the script automatically overwrites the previous version of the script.

- When you select File > Save for an unsaved script, or File > Save As for any graphical script, the ScriptChooser window opens. From this window, you can designate a file name and location for the script.

- For Script Author to be able to automatically recognize a script file, it must end with a .SCRIPT or a .SCR file extension. You can save a script without a file extension or with a different file extension, but you must explicitly change the file type default to All Scripts (*.*) in the ScriptChooser window to recognize file types other than .SCRIPT OR .SCR.

- Avoid naming a script with special characters (such as spaces, slashes, or backslashes) that are not permitted in the file system of some operating systems.

- Space characters are allowed, but not recommended. It is preferable to include other characters, such as an underscore (for example, script_name.script).

Guidelines for Saving to the Database

- When published, scripts automatically take their names and default language from

the global script properties. Scripts that have not been explicitly provided with a global script name (or for which a global language has not been selected) will default to a global script name of Untitled and a language of American. Therefore, you must provide your script with a global name and language before you publish it. Otherwise, you overwrite any previous untitled script designated for American English.

Reversing All Changes Since the Last Save to File System

Note: This feature is available only for graphical scripts saved to the file system, not to the database.

When you continually use the "Save" (and not the "Save As") command to save your script to the file system, Oracle Scripting enables two past versions of the file to be available for retrieval, in case you want to restore a previous version of the script.

The most recent version of the file is the "Save" version; the one saved prior to that is the "Backup" version.

Note: If you have both a "Save" and a "Backup" file, and use the "Save" command again from the Script Author:

- The old "Backup" file is overwritten by the previous "Save" file, which becomes the new "Backup" file.
- The newly-saved file becomes the new "Save" file.

Prerequisites

A version of the script must have been saved or backed up to the file system.

Steps

1. As you start to edit a script which has been saved previously to the file system, and you want to revert to a previous version of the script from the file system, you can select from the following menu options, when they are available:
   - File > Revert To > Last Save

     This restores the script to the last explicitly saved file.
   - File > Revert To > Last Backup

     This restores the script to the version of the file saved just *before* the last explicitly saved file.

Importing Scripts

When you import a script, the entire script - the root graph, and all subgraphs - is

placed into your current script, as a group, whose name is the script name of the imported script.

If you do not want all the imported objects in your script, you can delete them after the import.

> **Note:** Every Script Author script is customized. It is the script developer's responsibility to ensure that imported scripts function in the target environment.

> **Note:** If a script that you import contains commands referencing custom code (e.g., Java, PL/SQL, Forms, and so on), the corresponding code referenced by the script must be available in the environment in which the script is deployed.

> **Note:** *Do not assume imported scripts will function without thorough unit testing.*

Steps

1. Choose File > Import....
2. Using the Location field, if applicable, navigate to the local or network location from which you want to import the script.
3. In the File Type field, select Oracle CRM script file (.script).
4. In the Files window, select the script file name and then click Open.

   The imported script appears on the canvas as a group.

Exporting Script Groups as Separate Script Files

Exported groups can add modularity and code reuse to your script projects by enabling two separate approaches. When you export the group, you designate a file name and location on a local or network file system, and the group is saved as a separate script.

Thereafter, if you import it into another script, all script objects and commands associated with the original group are now part of the script into which they are imported.

Additionally, if you open the exported group in Script Author, the objects that were contained within the group appear on the root graph of the canvas.

> **Note:** Every script is a customized product. If a group is fully functional

when exported, this does not guarantee that the same functionality will be granted to a script into which it is imported without modifications.

Note: If a group that you export contains commands referencing custom code (e.g., Java, PL/SQL, Forms, and so on), the corresponding code referenced by the group must be available to the Scripting Engine in any environment into which the group is imported.

Note: *Do not assume exported scripts will function without thorough unit testing.*

Steps

1. On the canvas, select a group.
2. From the menu, select File > Export....
3. Using the Location field, if applicable, navigate to the local or network location to which you want to export the group, as a separate script.
4. In the File Name field, type an appropriate file name for the new script.

   Note: If you intend to import the script, provide a .SCRIPT file extension, so that the script is visible to the Open dialog without changing script file type settings.

5. Click Save.

   The Save window closes. The group is exported as a separate script with the file name you provided. The original script from which you exported the group remains open.

Guidelines

- The file name you provide to the exported script must be different from the global script name.
- When opening a script that was exported as a group, the group name becomes the global script name.
- Whether opening the exported group or importing it, any shortcut property in the original group is not retained in the exported script file.

Printing Graphs

Use this procedure to print a graph displayed on the Script Author canvas.

Steps

1. Select the graph you wish to print.
2. Choose File > Print.
3. Select your print options and then click OK.

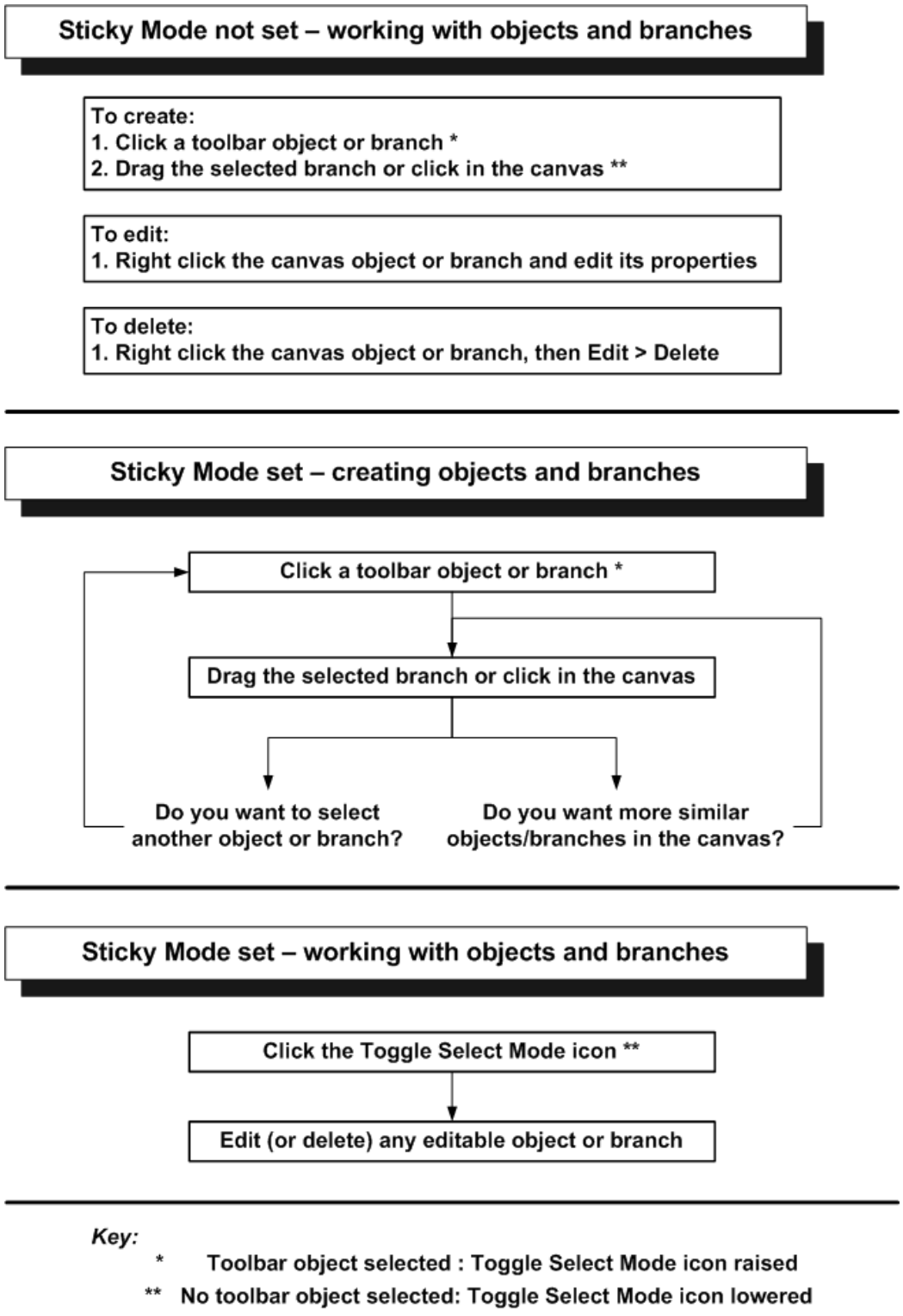
> **Note:** To print a subgraph contained within a group or block, you must first click to select the appropriate container object (the parent group or block) and then click the Go down into child graph button.

Enabling and Disabling Sticky Mode

The basic principle behind sticky mode is that it enables you to quickly create a number of canvas objects of the same type: you click an object toolbar button once, then click several times in the canvas to create many objects of the selected type.

In every Script Author session, the *sticky mode* feature is enabled by default. The only way to view, enable, or disable sticky mode is from the File Menu option.

A summary of how sticky mode affects Script Author operations appears in the diagram following.



Enabling and Disabling Sticky Mode

- Click the File menu option.

- Enable or disable sticky mode by respectively setting or unsetting the Sticky Mode checkbox.

Creating Objects and Branches in Sticky Mode

The basic principle behind sticky mode is that it enables you to quickly create a number of canvas objects of the same type, by minimizing the number of toolbar clicks. To do this, you must first select a canvas object in the toolbar. That object type remains "stuck" (selected) until you explicitly select another object or branch type from the toolbar.

Each time you click in the canvas, another instance of the "stuck" object type will appear. For example, if you select the panel object in the toolbar, then click three times in the canvas, three panel objects are inserted there.

If your selected object is a branch type, you can create - by dragging - many branches of the selected type in the canvas without re-selecting from the toolbar.

Working with Objects in Sticky Mode

When sticky mode is enabled, you can edit click canvas objects by right clicking them, but you cannot double-click them. To perform general editing and repositioning of canvas objects, you must first click the Toggle Select Mode icon in the toolbar. This deselects any previously-selected object, and you can now work directly on one or more canvas objects.

Working with Objects with Sticky Mode Not Set

When sticky mode is disabled, if you want to create a new canvas object, you must first select the corresponding toolbar icon before clicking in the canvas. You can edit or reposition existing objects, by working directly in the canvas.

Closing Scripts and Exiting Script Author

You can close a script or exit Script Author at any time. If you have unsaved changes, you will be asked whether you want to save the changes.

Steps

1. From the File menu, perform one of the following:
   - Choose Close to close the current script.
   - Choose Exit to leave Script Author.

     In either case, if you have made changes to the current script, you are asked if you wish to save the changes, before the script is closed.

Defining Global Script Attributes

Global script attributes are attributes which apply to the entire script. There are four types of global script attributes, as shown in the table below.

Global Script Attribute	Runtime Interface	Description
Script properties	All	Global script properties include the script name (as recognized by the database), comments, the script language, and boolean properties such as footprinting, answer collection, and suspendability.
Script pre- and post-actions	All	Commands that execute at the start or end of a script.
Script information area (defined in the Static Panel)	Agent	Displays a panel of information above the script at runtime in the Scripting Engine agent interface only. Data types include text and timers. Information can be about the script itself, or about the current interaction.
Shortcut buttons (defined in the Shortcut Panel)	Agent	Displays one or more functional buttons above the script at runtime in the Scripting Engine agent interface only. Buttons can execute any runtime command. Typical uses are as shortcuts (to progress the script to a specified point) or to launch a browser with a specified URL. Oracle Scripting APIs allow shortcut buttons to be dynamically enabled or disabled based on programmed events or conditions relevant to a script session.

Although not configurable from the Script properties dialog, the Disconnect button and the Suspend button can also be considered global script attributes.

When defined, global script attributes can affect or display data, or control the manner in which scripts appear at runtime.

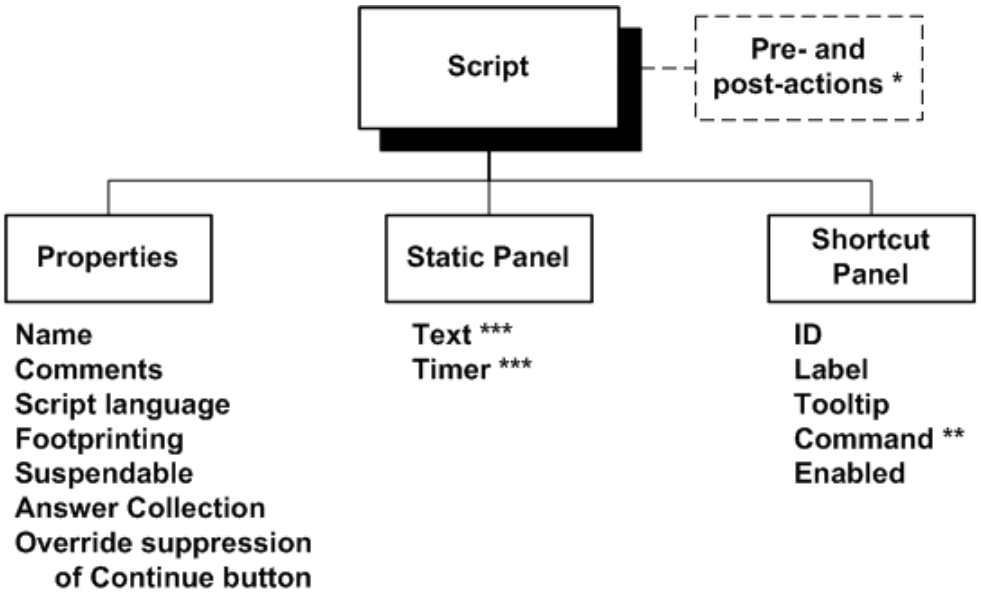
Some global script attributes (the Disconnect button, the Suspend button, script information area, and shortcut buttons) only appear at runtime to users of the Scripting Engine agent interface.

This section consists of the following topics:

Script Properties

Global script properties control the way scripts behave from a database perspective and at runtime, and affect data that is collected when scripts execute in the Scripting Engine.

The main components of a script that you define in the Script Author are shown in the following diagram.



Key:

* Java, PL/SQL, Blackboard, Forms, Constant Command, or Delete Action

** Java, PL/SQL, Blackboard, Forms, or Constant Command

*** Up to 9 Text and Timer elements in total

These script properties are all accessible in the Script Author by selecting File > Script Properties or by right-clicking in empty canvas space and selecting File > Edit Blob Properties.

For details of the pre- and post-actions, see Defining Actions, page 8-88.

Notes on Script Properties

Script Name

1. The global script name is distinct from the script name from a file system perspective. If you intend to use the same for both, ensure that you avoid using spaces, slashes, or backslashes, as these characters are not permitted in file names of some operating systems. Using a .SCRIPT or .SCR file extension in the file name is advisable, but not mandatory.

2. When a script is created, it is automatically designated a global script name of "Untitled1." Unless you assign a new global script name, deploying it to the database will overwrite any other unnamed scripts that have been deployed to the database with the same default name and script language properties.

Script Language

1. The Script Author script language setting can be any uni-drectional language

supported by Oracle Applications in the FND_LANGUAGES table. The default value is AMERICAN.

2. Changing the script language setting does *not* translate the GUI elements nor the script elements.To produce a version of a script in another language, you should do the following:

    1. Change the Script Name and Script Language.

    2. Manually translate all panel text, answer lookup values, panel names or labels, or any customized aspect of the script.

    3. Save and deploy the new version of the script.

Footprinting

Footprinting is the recording in the Oracle Applications database of the names of each panel in a script transaction that is visited during a script transaction, and the duration of time (in milliseconds) prior to the activation of the next panel.

For more details, see Global Script Properties, page 2-14.

Answer Collection

1. When you enable answer collection at the script properties level, Oracle Scripting, by default, records the responses to questions in any panel provided at runtime by script end users of any Scripting Engine interface.

2. You can override answer collection for individual questions, as follows:

    - From the Answer Entry dialog for a specified answer, Edit Data Dictionary > General Tab > Uncheck the Collectable? checkbox.

For more details, see Global Script Properties, page 2-14.

Suspendable

You can allow scripts to be suspended and resumed, in both of the Scripting Engine interfaces.

For more details, see Global Script Properties, page 2-14.

Override Suppression of Continue Button

By default, any Continue buttons created in script panels are suppressed when the script executes in the Web interface. If you want to keep the Continue buttons visible in the Web interface, you can override this suppression.

This parameter does not affect the appearance of scripts executed in the agent interface.

Static Panel

1. The Static Panel defines a script information area that displays, the agent interface only, above each panel at runtime. The script information area can display up to nine elements, each of type Text or Timer.

    For more details, see Global Script Properties, page 2-14.

Define the following fields for text and timer elements:

ID: This represents the internal identifier of the element; this is required for Oracle Scripting APIs

Label: An optional field, this contains the label that you want to appear in the script information area at runtime

Command:

For a text element, either define a Constant or a Blackboard command, with the value you want to appear at runtime designated as the return value of the command.

For a timer element, you must define a Java method that invokes the **startTimer** Oracle Scripting API. To do this, you must *leave the timer element Command blank*, and associate a timer command that references the timer ID to one or more shortcuts or to a pre- or post-action for the script or for any valid script object.

For more information, see the topic *Defining the Script Information Area* in the *Oracle Scripting Developer's Guide*.

Shortcut Panel

1. If one or more shortcut buttons are defined and enabled in the Shortcut Panel, then they appear at runtime in the Scripting Engine agent interface immediately above the panel display area, and below the script information area.

   For more details, see Global Script Properties, page 2-14.

2. When you add a shortcut button in the Shortcut Panel, define the following fields in the Shortcut Info Entry Dialog window

   ID: This represents the internal identifier of the shortcut

   Label: This contains the label that you want to appear on the shortcut button

   Tooltip: This contains the Help text that you want to appear when an agent's pointer hovers over the shortcut button.

   Command: This contains the command to be executed when an agent clicks the shortcut button.

   Enabled: When checked, this enables the shortcut button when the script is launched.

   > **Note:** Defining a *shortcut button* is not the same as defining a *shortcut*.

   > **Note:** A shortcut is a group property, and can be used to associate a target destination for a shortcut button, but shortcuts have other uses also.

Note: For more information, see Defining Shortcuts, page 8-68.

For more information on setting up shortcuts and shortcut buttons, see the section Performing Advanced Graphical Script Tasks of the in the *Oracle Scripting Developer's Guide*.

Programming the Script Disconnect Button

Although not configurable from the Script properties dialog, the Disconnect button can also be considered a global script attribute.

Use this procedure to program the Disconnect button that appears in the Scripting Engine agent interface. Clicking the Disconnect button at runtime routes the agent directly to this group. The group need not contain any panels, as long as it meets the syntax rules for any group (see Minimum Requirements for Any Graph). If panels are included in this group, they will be displayed each time the Disconnect button is clicked. For a quick exit disconnect feature, simply insert a termination node into this group and attach the start and termination nodes with a default branch.

Prerequisites

- Insert a group on the root graph, page 8-67.
- Provide proper termination for the group.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a group.

   The Properties window appears.
3. In the Group tree, select Shortcut.

   The Shortcut pane appears.
4. In the Shortcut pane, in the Shortcut field, enter the value WrapUpShortcut.

   This must be entered exactly as indicated (this property is case sensitive).
5. Click Apply to apply your work or click OK to save your work and exit the Properties window for the group.

   When the Disconnect button is selected at runtime, the first panel of this group (if any) displays.

   The objects in the group (if any) and commands associated with them (if any) are executed, each in sequence, until the termination node is reached

   Once the termination node on the root graph is reached in the processing of the script, than any global script post-actions execute, and the script session at runtime

terminates.

References

- Minimum Requirements for Any Graph, page 2-27

Working with Panels

The panel is the only Script Author object that is visible at runtime. The composition of each panel is determined by script developers and script requirements. Panels may contain panel text, images, embedded values, hypertext links, customized HTML tables, and any number of questions and their associated question user interface controls.

Panels may include any number of questions (there are no limits other than practicality).

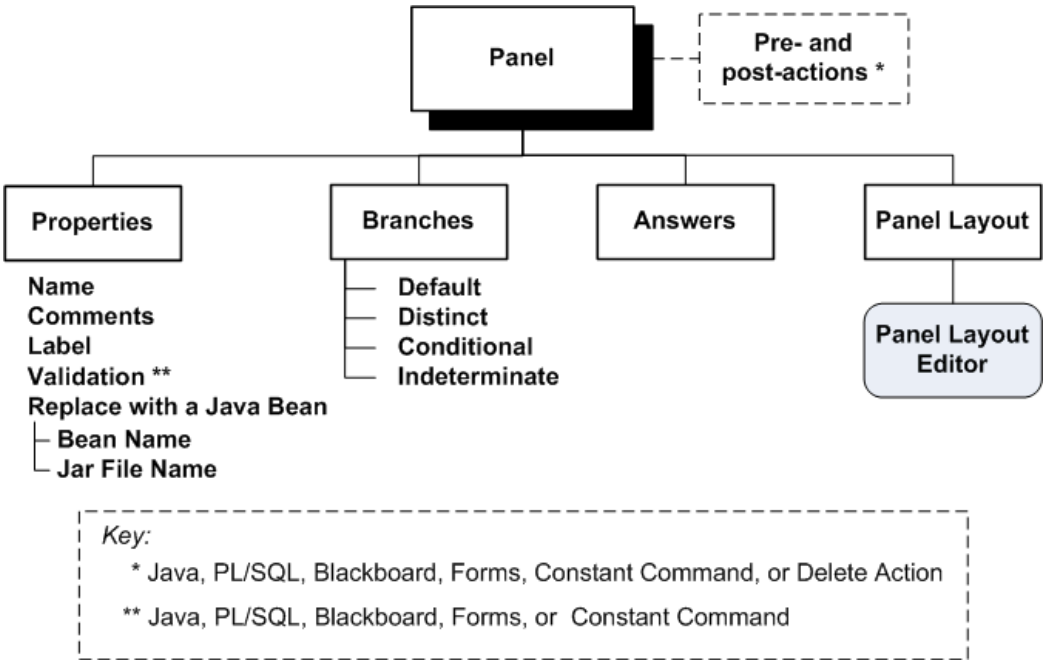
This section consists of the following topics:

- Panel Properties and Attributes, page 8-29
- Notes on General Panel Properties, page 8-30
- Inserting Panels, page 8-31
- Defining General Panel Properties, page 8-31
- Button Considerations for Panels, page 8-31
- Substituting a Java Bean for a Panel, page 8-32

Panel Properties and Attributes

A panel contains the information that is displayed in the script at runtime. Answer controls defined in the panel (using the Script Author) are used to accept information from the user at runtime and advance the flow of the script.

The main components of a panel that you define in the Script Author are shown in the following diagram. The attributes and elements in the diagram are the panel properties that you can view and edit.



These panel properties and attributes are accessible in the Script Author by right-clicking a canvas panel and selecting File > Edit Blob Properties, or if you have enabled automatic properties popup when creating a new panel, group, or block.

For more information about actions and branches, see:

- Defining Actions, page 8-88
- Working with Branches, page 8-82

For information about properties of questions and answers, see:

- Understanding Answer Properties and Answer Types, page 8-33
- Working with Answers and Answer Properties, page 8-44

For information about the panel contents - the text and answer controls that define what appears in the panel at runtime - see Working with the Panel Layout Editor, page 8-50.

Notes on General Panel Properties

Name

Identifies the panel in the Script Author and the database. Also used to track a panel for footprinting purposes.

Each panel is automatically provided with a panel name unique to the current script. Overwrite this value in the Name field with the desired name for the panel.

Label

Identifies the panel at runtime in the Progress Area of the Scripting Engine agent interface.

Comments

Optional descriptive text.

Validation

You add a command here to validate the contents of a panel. The validation occurs as you exit the panel.

Replace with a Java Bean

For details, see Substituting a Java Bean for a Panel, page 8-32.

Inserting Panels

The panel is the only Script Author object that visible at runtime, potentially including the display of panel text, question controls (and, optionally, associated labels), graphics, hyperlinks, or embedded values.

For general information on inserting panels, see Inserting Objects and Branches, page 8-9.

For details of panel properties, see Panel Properties and Attributes, page 8-29.

Notes

1. Ensure you define at least one question for each panel you insert, or the script will not pass a syntax check. In runtime, every panel requires end user interaction (an answer to a question).

Defining General Panel Properties

The basic process of defining the general panel properties is as follows:

1. On the canvas, right-click the panel > Edit Blob Properties.
2. Click Properties.
3. Enter or edit the general panel properties.

   For details, see Notes on General Panel Properties, page 8-30.

Button Considerations for Panels

When displayed at runtime, each panel includes a Continue button. At runtime, script end users must click Continue to progress through the script.

Each panel must contain, at minimum, a single question user interface control. In a graphical script, you must explicitly define each question control. (Wizard scripts can

automatically insert a Continue button in a panel, if there are no other questions defined.)

When you define a single button question control in a panel, you can provide any value to the button. Oracle recommends using the value *Continue* (for the value and the display value for the button control). This mimics the button that progresses panels with other question types. If you set the display value to any text string other than *Continue*, then you can consider this UI control to be the one exception to the rule that each script includes a Continue button.

You can also provide several answer definitions (or lookup values) for a button control. Each displays as a separate button. Clicking on any one at runtime registers that as the end user's selection and progresses the script to the next object, displaying the next panel in sequence. Buttons are designed to appear in a single horizontal row. However, when button values are too long, or if there are many answer choices, or if the window size at runtime is small, the buttons wrap to successive horizontal lines in the panel, displaying in as horizontal rows as is necessary. There is no way to modify this behavior.

If a panel includes more than one question UI control, then you cannot define a button for the panel. However, a Continue button will be automatically generated by the Scripting Engine at runtime for the panel.

Substituting a Java Bean for a Panel

Use this procedure to substitute a Java bean for a panel. In order for the Java bean code to execute, the script must appropriately reference the bean, the bean must be appropriately packaged, and deployed to the applications database.

> **Note:** Java beans are customized components. As such, no support is provided for scripts that have difficulties substituting panels with Java beans. The functionality is provided with the tool to allow unsupported customization.

Prerequisites

- Insert a panel onto the canvas, page 8-31.
- Write the code for the Java bean.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a panel.

   The Properties window appears.

3. In the Panel tree, select Properties.

   The Properties pane appears.

4. In the Properties pane, select Replace with a Java Bean.

   The Bean Name and Jar File Name fields are enabled.

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 window.

References

- Packaging Java Bean or Custom Java Code Into a JAR File, page 8-106

Understanding Answer Properties and Answer Types

> **Caution:** In Oracle Scripting, there is a one-to-one relationship between questions and answers. In this section the terms "question" and "answer" are used interchangeably, with the most appropriate term used in context.

This section consists of the following topics:

- Answer Properties, page 8-33
- Answer Properties Specific to Answer Types, page 8-39

Answer Properties

You can view and edit answer properties when you add or edit an answer in the Panel Properties window of a panel.

The main properties of each answer appear in different windows, as follows:

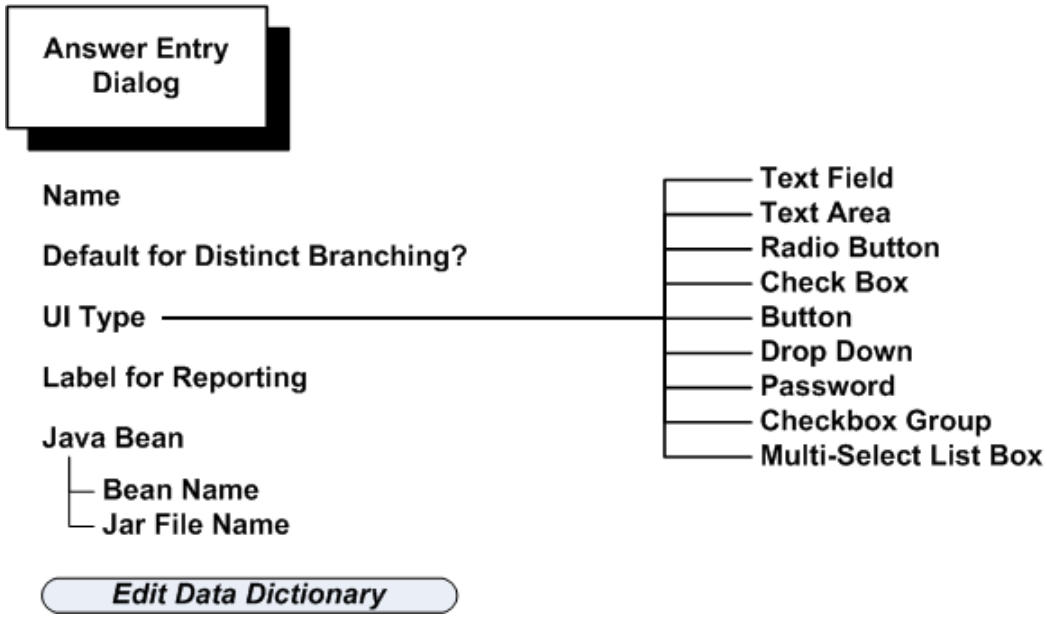
- Answer Properties in the Answer Entry Dialog Window, page 8-34
- Answer Properties in the Edit Data Dictionary Window, page 8-36

See Also

- Answer Properties Specific to Answer Types, page 8-39
- Working with Answers and Answer Properties, page 8-44

Answer Properties in the Answer Entry Dialog Window

The following diagram shows the answer properties that appear in the Answer Entry Dialog window:



From the Answer Entry Dialog window, more answer properties are available when you click the Edit Data Dictionary button; they appear in the Edit Data Dictionary Window, page 8-36.

Notes on Answer Entry Dialog Properties

The following table lists the answer properties in the Answer Entry Dialog window.

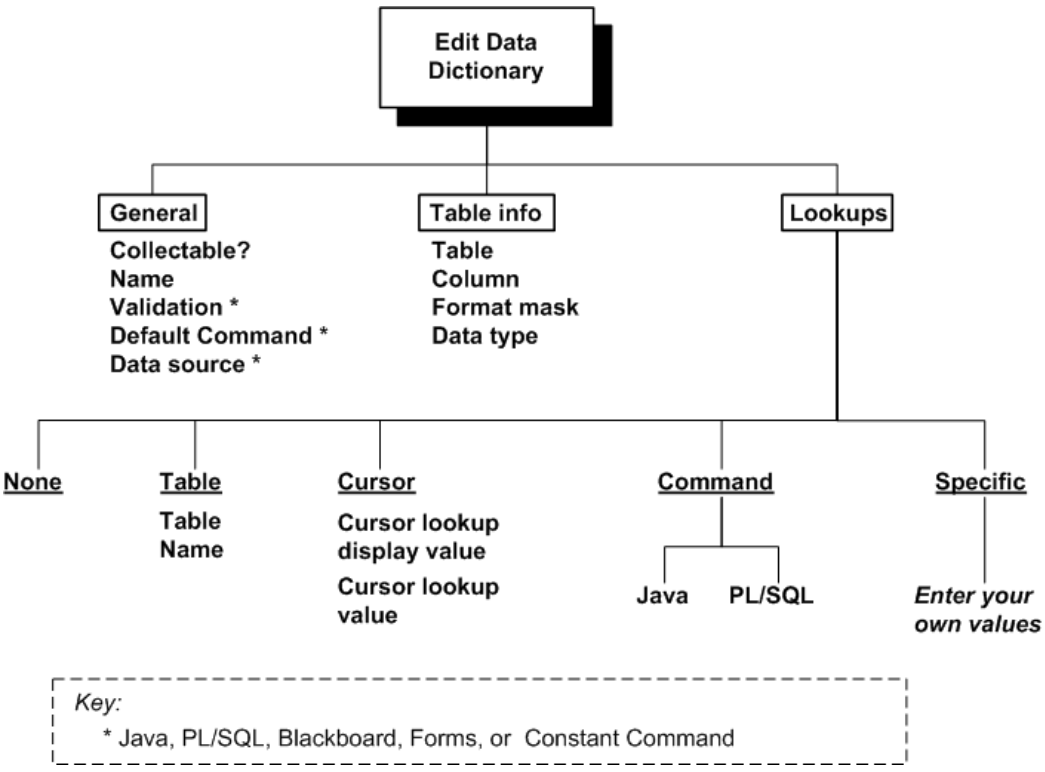
Property	Data Type	Required?	Function or Description	Restrictions
Default for Distinct Branching	Check box	*(See Restrictions column.)*	**Note: Not an option if UI Type is Checkbox Group or Multi-Select List Box.** Indicates whether this question will be triggered as the default for distinct branching at runtime.	Required only when distinct branching is used as an outgoing branch for a panel. Only one question per panel may have this option selected.

Property	Data Type	Required?	Function or Description	Restrictions
Name	Text	Yes	Identifies the specific question to the Oracle Scripting application. When a value is provided at runtime, this value is the key to access the response value provided for this question by the application end user.	Supports any text in the current character set. For each panel, the panel name should be unique.
UI Type	Drop-down List	Yes (defaults to Text Field UI type if no selection made)	Identifies the answer control which renders at runtime for end user input. Choices include Text Field, Text Area, Radio Button, Check Box, Button, Drop Down, Password, Checkbox Group, and Multi-Select List Box.	One UI type per question.
Label for Reporting	Text	Only for single Check Box UI	For all UI types except check box, entering text in the label for reporting results in that text string appearing as a label to the left of the answer at runtime. This label also appears in reports generated for the application.	Check boxes will display this value to the right of the single check box, instead of the lookup's display value.
Bean Name	N/A	No	This field, formerly used to identify a Java bean name with which to replace a question, is no longer supported due to conflicts with the WYSIWYG architecture.	Not functional with Oracle Scripting 11.5.6 and later.

Property	Data Type	Required?	Function or Description	Restrictions
Jar File Name	N/A	No	This field, formerly used to identify a source code Java archive for a Java bean to replace a question, is no longer supported due to conflicts with the WYSIWYG architecture.	Not functional with Oracle Scripting 11.5.6 and later.

Answer Properties in the Edit Data Dictionary Window

The Edit Data Dictionary window displays three tabs: General, Table info, and Lookups, each of which has its own properties, as shown in the following diagram.



Notes on Edit Data Dictionary Properties

Notes on Edit Data Dictionary Properties

General tab

Collectable?

The script property Answer Collection sets the general rule for a script. You can override this general rule for individual answers with the Collectable? property.

Name

Identifies the answer.

Validation

Specifies a command to perform validation on the answer.

Default Command

Add a command to provide the default answer at runtime. At runtime, the default answer displays initially and when you click Reset to Default.

Data source

A data source is code that pulls data from or sends data to an external application, and is executed when you exit the panel. Oracle recommends that you use the equivalent functionality available through panel post-actions

Table info tab

Table info columns (for answers in Insert and Update block subgraphs)

The Table info area can be used for an answer that belongs to a panel in the subgraph of an Insert or Update block: if you specify Table Name and Column Name, the value of the runtime answer is stored into or replaces the named table and column when the block insert or update is performed.

Table info columns (for answers in Query block subgraphs)

1. The Table info area can be used for an answer that belongs to a panel in the subgraph of a Query block: if you specify Table Name and Column Name, this information, together with the value of the runtime answer, is is used as a query constraint when the block query is performed.

   For example:

   - The answer name is OrderQty, and is a Text Field.
   - The table for the query block is Order_Items, with a column Qty.

   You want to restrict the query search results to those rows whose quantity is greater than the value entered for the OrderQty answer.

   - In this case, for the OrderQty answer, in the Table info area, specify TableName = Order_Items, Column Name = Qty.

   If no other query constraints are defined, either in the subgraph or the Object Dictionary of the query block, Oracle Scripting sets up a query of the form:

   ```
   SELECT ... from Order_Items where Qty = : OrderQty
   ```

2. You can use the Format mask in the Table info data to convert queried date columns, as in the following example:

- The answer name is EnteredDate, and is a Text Field.
- The table for the query block is Order_Items, and it has a column Order_Date whose column type is Date.
- In the script, the user enters a value of ''12-DEC-2006' for the EnteredDate answer. Oracle Scripting treats this as a string of characters.

To enable the script to retrieve all orders for that date, in the Table info area, you must define Table Name = Orders, Column Name = Order_Date, and Format Mask = DD-MON-YYYY.

Oracle Scripting will generate a query similar to the following:

```
SELECT ... from Orders
where to-char(Order_Date, 'DD-MON-YYYY') = '12-DEC-2006';
```

Lookups tab

Table lookups

You can set up a custom table to be used to retrieve lookups. The table must contain the exact columns as shown and described in the following definition:

Column	Type and Size	Notes
ANSWER_ID	NUMBER	-
ANSWER_VALUE	VARCHAR2 (512)	Answer value that is used internally in Oracle Scripting.
ANSWER_DISPLAY_VALUE	VARCHAR2 (512)	Answer value that is displayed in a script.
ANSWER_ORDER	NUMBER	Sequence of answers when displayed in a script in the agent interface.
ANSWER_ACTIVE	NUMBER	If set to 1, the row is displayed as a lookup choice, else the row is not displayed

Cursor lookups

Cursor lookups derive from the results of the last database query block executed .

Cursor display lookup value

Actual text shown to the end user to select.

Cursor lookup value

Value passed by the application when selected at runtime.

Command lookups

Use a Java or PL/SQL command to provide the lookups.

Specific lookups

Enter your own lookups. Specify both the Display Value (the text shown at runtime) and the Value (used by the application)

Answer Properties Specific to Answer Types

This section consists of the following:

See Also

- Answer Properties, page 8-33
- Working with Answers and Answer Properties, page 8-44

Characteristics of Specific Answer Types

Script Author provides nine answer types, consisting of a familiar set of answer controls that render in HTML forms: buttons, checkboxes, radio buttons, text fields, text areas, and password fields. However, some of the characteristics of each may act differently than in a standard HTML page. The table below briefly describes each type.

Answer Type	End User Action	Supports Null Value?	Supports Multi-Select Values?	Requires Lookup Values?	Requires Label for Reporting?
Text Field	Keyboard entry	Yes	No	No	No
Text Area	Keyboard entry	Yes	No	No	No
Radio Button	Click	No	No	Yes	No
Check Box	Click	Yes	No	No	Yes
Button	Click	No	No	Yes	No

Answer Type	End User Action	Supports Null Value?	Supports Multi-Select Values?	Requires Lookup Values?	Requires Label for Reporting?
Drop Down	Click, drag, click	No	No	Yes	No
Password	Keyboard entry	Yes	No	No	No
Checkbox Group	Click	Yes	Yes	Yes	No
Multi-Select List Box	Click, Ctrl-Click (Option-Click for Macintosh OS)	Yes	Yes	Yes	No

Usage and Sharing of Answer Characteristics

Each answer behaves according to its own specifications. This section describes how characteristics are used and shared by the various answers.

Questions Where You Enter Data at Runtime

Text field, text area, and password field questions all accept keyboard entry from the user. While the answer control differs in appearance or behavior for each, each type will accept up to 4,000 characters.

Questions Where You Select Data at Runtime From Defined Lookups

1. Radio buttons, check boxes (both individual and checkbox groups), buttons, and dropdown lists (single and multi-select types) all require you to perform an action at runtime (a mouse click or a keyboard command) to select an answer (assuming no default command is defined for the answer). The script designer must define lookup values for these questions.

2. Radio buttons and dropdown lists require you to select an answer at runtime before you click the default Continue button.

Questions Where Null Values are Accepted at Runtime

Single checkbox and both multi-select answer types (checkbox groups and multi-select list boxes) can accept null (unselected) as a valid response.

Questions Where Label for Reporting is Required

Only the single checkbox answer type requires a label for reporting. For any reports

generated, this value appears for checkbox question types in the report. At runtime, in a panel, this label is displayed to the right of the checkbox. In contrast, values which appear at runtime for checkbox groups are answer choices (or lookup values).

Questions Where Label for Reporting Does Not Appear on Panel

Only the button answer type does not display any value entered into the Label for Reporting field. In reports generated, this value still appears.

See Also

- Guidance on Using Answer Types, page 8-41
- Working with Answers and Answer Properties, page 8-44

Guidance on Using Specific Answer Types

Guidance on when to use specific answer types follows below, along with answer type-specific information and recommendations.

Answer Type	Usage	Design Factors	Runtime Factors
Text Field	Use when soliciting a short text response at runtime.	Default answer type.	You can enter and scroll through text that is longer than the text box space.
Text Area	Use when soliciting large amounts of text at runtime, with space for inputting several lines.	Labels are optional, but are recommended, for clarification or if you have several text input controls in the panel.	You can enter and scroll through text that is longer than the text area space.
Radio Button	Use for a series of conditions for which only one is allowed.	Requires lookup values. Labels are optional, as the function of this answer type is generally evident.	You must select one of the options.
Check Box	Use to evaluate a simple "true/false" condition.	You must specify a value for the Label for Reporting. Do not specify lookup values.	At runtime, select the check box if the condition is true, otherwise leave it blank.

Answer Type	Usage	Design Factors	Runtime Factors
Button	Use to progress the script to a panel.	**A panel with the Button answer type cannot contain any other questions.** A panel with the Button answer type will not contain the automatically generated "Continue" button that appears with all other answer types. Requires lookup values. Can have one or more buttons in the panel. Single button: The lookup value in this case is typically "Continue." Multiple buttons: Each lookup value becomes button text at runtime. If Label for Reporting is defined, it is used in generated reports, but does not appear in the panel at runtime.	Single button: You direct the script to a specific panel, often the next panel. Multiple buttons: You select a button from the horizontal row of buttons to direct the script to a specific panel.
Drop Down	Use for a series of conditions for which only one is allowed. *Similar to Multi-Select List Box, except Drop Down does not support null values*.	Requires lookup values. Labels are optional, as the function of this answer type is generally evident.	You must open the drop down field and select a value.

Answer Type	Usage	Design Factors	Runtime Factors
Password	Use for short amounts of text to be entered, with entered data appearing as asterisks.	Labels are optional, but are recommended, for clarification or if you have several password input controls in the panel. Oracle recommends that you specify the Validation property in the question's data dictionary (General tab): you enter a command, for example, to verify the number or format of characters input, or to compare a user-supplied password against a validation table.	If you enter text beyond the allocated space, you can scroll across, but all the data appears as asterisks.
Checkbox Group	Use for a series of conditions for which zero, one, or many selections are allowed.	Requires lookup values, which at runtime will appear beside their related checkbox. Label for Reporting, if defined, appears at runtime as a label to the left of the checkboxes and lookup values. Default for Distinct Branching property is disabled, as users can select more than one value.	Each check box appears with its corresponding lookup value to the right, in a horizontal row, which wraps if enough room is not available. You can leave all the checkboxes blank, to enable the script to continue to the next panel.
Multi-Select List Box	Use for a series of conditions for which zero, one, or many selections are allowed.	Requires lookup values. For the ability to select multiple values, you should define more than one lookup value, even though a single lookup value is syntactically correct. Default for Distinct Branching property is disabled, as users can select more than one value.	Up to three lookup values are visible, with a vertical scroll control to display more if applicable. To select multiple values, hold down the Control key and click once on each desired selection. To select multiple sequential values, you can use the Shift key.

See Also

- Characteristics of Specific Answer Types, page 8-39
- Working with Answers and Answer Properties, page 8-44

Working with Answers and Answer Properties

> **Caution:** Oracle Scripting, there is a one-to-one relationship between questions and answers. In this section the terms "question" and "answer" are used interchangeably, with the most appropriate term used in context.

This section consists of the following tasks related to answers and answer properties:

- Adding and Editing Panel Answers, page 8-44
- Defining Lookups, page 8-45
- Using Answers in Table Insertions and Updates, page 8-46
- Adding Validation to Answers, page 8-46
- Continue Button Considerations for Answers, page 8-48
- Restrictions for Answer Property Changes, page 8-49

Adding and Editing Panel Answers

Answers are properties of a panel.

The basic process of defining an answer is as follows:

1. Right click a canvas panel, then select Edit Blob Properties
2. Click Answers, then click Add or Edit as appropriate.

Starting in the Answer Entry Dialog window, and continuing in the Edit Data Dictionary window, you can then enter and select answer properties, as described in Answer Properties, page 8-33.

Reordering and Deleting Panel Answers

When you specify panel answers, you can control the order in which they appear at runtime. You can also delete answers from your list of answers for a panel.

The basic process is as follows:

1. Right click a canvas panel, then select Edit Blob Properties.

2. Click Answers.

3. Select an answer, then click Move Up, Move Down, or Remove as appropriate.

Defining Lookups

For certain answer types, you must define lookups, in order that users can select a value at runtime, for example, a value from a dropdown list, or a specific button. For details, see Answer Properties Specific to Answer Types, page 8-39.

The basic process is as follows:

1. Access the Answer Entry Dialog window, page 8-44.

2. Click Edit Data Dictionary.

3. Click the Lookups tab.

    1. To obtain answer choices from a specific table, select **Table lookup** and define the lookup table name.

        This option requires you to define the table in your database as described in Lookups tab, page 8-38

    2. To obtain answer choices from the Scripting cursor, select **Cursor lookup** and define the display value and the lookup value.

        This option requires a query to have been performed previously in the script.

    3. To obtain answer choices as return values to a command, select **Command lookup** and define the Java or PL/SQL command to execute that will return the desired values.

    4. To hard-code specific values as answer choices, select **Specify lookups** and then click Add to add one or more of the following pairs of data that describe each lookup:

    5. Display Value: this appears in the script panel at runtime.

    6. Value: this is stored in the Oracle Scripting database schema when the corresponding answer value is selected at runtime.

Reordering and Deleting Specified Lookup Values

When you specify lookup values, you can control the order in which they appear at runtime. You can also delete lookups from your specified list of lookups.

The basic process is as follows:

1. Access the Answer Entry Dialog window, page 8-44.

2. Click Edit Data Dictionary.

3. Click the Lookups tab, then select Specify lookups.

4. Select a lookup, then click Move Up, Move Down, or Remove as appropriate.

Using Answers in Table Insertions and Updates

You can define answers to be written to a database table, as part of the insert or update defined in an insert or update block.

Define this feature, which is available for answers in panels defined in the subgraph of an Insert or Update block, as follows:

1. Access the Answer Entry Dialog window, page 8-44.

2. Specify the Table Name and Column Name into which the runtime answer will be stored when the block insert is performed, or which column will be updated when the block update is performed.

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

You can define answers to be written to a database table, as part of the insert or update defined in an insert or update block.

Define this feature, which is available for answers in panels defined in the subgraph of an Insert or Update block, as follows:

1. Access the Answer Entry Dialog window, page 8-44.

2. Specify the Table Name and Column Name into which the runtime answer will be stored when the block insert is performed, or which column will be updated when the block update is performed.

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

Adding Validation to Answers

Answer validation establishes and enforces business rules for a designated answer in a panel. When the script end user provides a response to the corresponding question at runtime, the Scripting Engine validates the answer provided. If the answer does not meet the criteria specified in validation, then when the user attempts to exit the panel, an error results, and the user is prompted to change the answer before exiting the panel.

You can enforce the answer to be not null, or in a specified range of numbers, in a

particular format, or whatever other conditions controlled by the validation command. You can use custom or previously existing validation routines in your script.

Prerequisites

- If using custom code, write the code for the validation command.
- If using existing best practice Java methods, identify the appropriate Java method and its required parameters.

Steps

The basic process of adding validation to an answer is as follows:

1. Access the Answer Entry Dialog window, page 8-44.
2. Click Edit Data Dictionary.
3. In the General tab, click Edit beside the Validation field.
4. In the Command field of the Command Properties window, define a Java command for the validation action. For more details, see Defining Commands, page 8-89.

   Ensure you include any required parameters.

Using Custom Code

You can define custom validation for an answer by coding a Java method to meet your project requirements, and making this method available to Oracle Scripting. Like all custom code, this is exposed to the Scripting Engine by packaging the appropriately tested and packaged compiled class into a JAR or ZIP file as appropriate. The Java archive is then uploaded to the applications database from the Administration tab of the Scripting Administration console. You must either specify the custom code as global, or map the Java archive to your script.

Using Best Practice Question Validation Java Methods

You can also use existing validation routines available to Oracle Scripting. This requires no custom code. As detailed in *Oracle Scripting Developer's Guide,* there is a set of best practice Java methods (in the class NodeValidation) that ships with Oracle Applications. When associated as a Script Author Java command for question validation (in the data dictionary for the question), these can be used to validate questions in a script at runtime.

The best practice Java methods include validation that the answer provided to the appropriate question in a panel at runtime meets one of the following criteria:

- Is a valid date in a specified format (MM/dd/yyyy)
- Is a date in the future, in a specified format (MM/dd/yyyy)

- Is a valid time in a specified format (hh:mm:ss am/pm)
- Is an integer
- Is an integer in a specified range
- Is not null (a response is required)
- Is an invalid value (based on a parameter passed to the command)

Guidelines

- When defining the Script Author Java command, you must provide the fully qualified class path of the class and method you are referencing.

  For example, if using the method validateRequiredField in the best practice class NodeValidation, then in the Command Info area, in the Command field, type oracle.apps.ies.bestpractice.NodeValidation::validateRequiredField.

- Most of the best practice Java methods take parameters. Refer to the appropriate section of *Oracle Scripting Developer's Guide* for the list of best practice methods and the parameters required.

  For example, if using the method validateRequiredField, two parameters are required: a string called answer and a string called label.

  - The answer parameter defines the name of the panel answer for which you want the validation routine to apply. In the Parameter window, in the Name field, type the answer name.
  - The label parameter defines the label of the answer to appear at runtime in the message window. For example, if validating a field called Customer Number, then in the Parameter window, in the Name field, type label, and in the Value field, type Customer Number. At runtime, if the user attempts to pass this field without providing a value, then a dialog box labeled Message appears, with the message: *Please enter a value for: Customer Number.*

References

Oracle Scripting Developer's Guide

Continue Button Considerations for Answers

Panels with Two or More Answers Include Generated Continue Button

When displayed at runtime in the agent interface, each panel with *two or more explicitly defined answers* also displays a Continue button. At runtime, you must click the Continue button to progress through the script. Whether developing a script using graphical tools or the Script Wizard, no steps are required to generate this button. It is a

function of the Scripting Engine.

Automatic Question Creation for Wizard Script Continue Buttons

For wizard scripts, if you do not specify a question control in a panel, Oracle Scripting automatically creates a question, which displays a Continue button at runtime. The question is named Continue, and has a label for reporting of Continue. In the data dictionary for this question, the specified lookup value is also named Continue (for both the value passed to the application, and the display value).

Explicit Question Creation for Graphical Script Continue Buttons

For graphical scripts, even if a panel has no requirement for a text type or selection type question control, the script still requires a mechanism to progress to the next panel at runtime. Thus, *you must explicitly define a Continue button.*

Use the Button question control, and specify a question name (for example, Continue). Optionally, you can include a label for reporting.

You must also define, in the data dictionary for the question, a single answer choice (referred to in the graphical UI as a lookup entry) with a value and display value of Continue. To do so, in the question data dictionary, select Specify lookups in the General tab and click Add to access the Lookup entry window.

Restrictions for Answer Property Changes

1. Once the answer properties are saved, changes to Answer properties behave in the following manner in Script Author:
   - Changes to the Default for Distinct Branching option *will* be recognized, unless the UI Type is one of the multi-select options.
   - Changes to the Name property *will* be understood by the Scripting Engine regardless of execution interface.
   - The UI Type list is *not* modifiable from the Answer Entry Dialog window.
   - When you first enter a value in the Label for Reporting field, when you save the changes by clicking OK, a corresponding label is generated in HTML panel text. When you return to the Answer Entry Dialog window and make changes to this value:
     - The new value is updated in the IES schema.
     - The new value appears in subsequent reports showing panel questions and labels.
     - The new value displays in the Answer Entry Dialog window.

- The new value *will not be reflected in the panel layout*. You must also edit this value in panel layout HTML manually to reflect changes in the Label for Reporting field.

If you want to change either the UI type or the Label for Reporting property of the Answer Entry Dialog window, you must take one of following approaches:

1. Export the panel HTML. Customize the panel HTML code to modify the UI type or Label for Reporting, following appropriate panel and UI type syntax rules. From Script Author, re-import the panel text, replacing the previous question of the same name.

2. Create a new question, applying the appropriate properties. If you wish to provide the newly defined question with the same question name, you must delete the original question prior to checking script syntax, or you will encounter the error message "error: duplicate answer name <question name>".

3. For the Label for Reporting property only, you can change the value in the Answer Entry Dialog window, and also manually edit the panel layout, either using the panel layout editor or by modifying the HTML code.

   Note: If you do not change the Label for Reporting value both in the Answer Entry Dialog window *and* in the panel layout, the values will not be in synch.

2. For scripts built for execution in Oracle Scripting 11.5.6 or later, do not fill in any parameters in the Java Bean area of the Answer Entry Dialog window. Replacement of questions within a panel with a Java bean is no longer supported due to conflicts with the WYSIWYG panel rendering architecture.

   Note: You can still replace an entire panel with a Java bean. Be aware that doing so is considered unsupported customization. The Java bean is recommended to be fully tested. Java code must be appropriately packaged and fully qualified, and must be deployed to the database. You can deploy custom Java archives in JAR or ZIP format by navigating to Administration tab > JarListings subtab using the Scripting Administration console.

Working with the Panel Layout Editor

The panel layout editor is a feature available to graphical scripts through which you can add, modify, and view panel text, apply formatting to text and panel questions, insert graphics, create hypertext links, and add embedded values to display dynamic content.

You can also export panel HTML and import modified or custom panel HTML.

This section consists of the following topics:

When Should You Use the Panel Layout Editor?

Each panel in a script displays at least one answer at runtime, and may contain other elements such as formatted text and graphics.

The appearance or of each panel at runtime is controlled by panel layout HTML, generated in Script Author either by the Script Wizard, or by Script Author graphical tools.

You should first define your answers and their properties in the Answer Entry Dialog window. These answers defined for a panel will then automatically appear in the panel layout.

You can then create, modify or add optional panel content in a graphical script using the panel layout editor.

What Can You Do in the Panel Layout Editor?

In the Panel Layout Editor, you can enter text and then format it in several ways:

- *As a convention, you can choose to indicate that text has two special usages at runtime. Instructionaltext*, which displays as dark blue text,indicates specific instructions to

the script end user. *Spoken text*, which displays as magenta text, indicates text that an agent would speak (using the agent interface) or that a customer or survey respondent would read for communication of primary information for that panel. Oracle Scripting does *not* enforce the intended usage indicated by the special color fonts.

- You can also select from a variety of typeface styles, sizes, justification and formatting options.

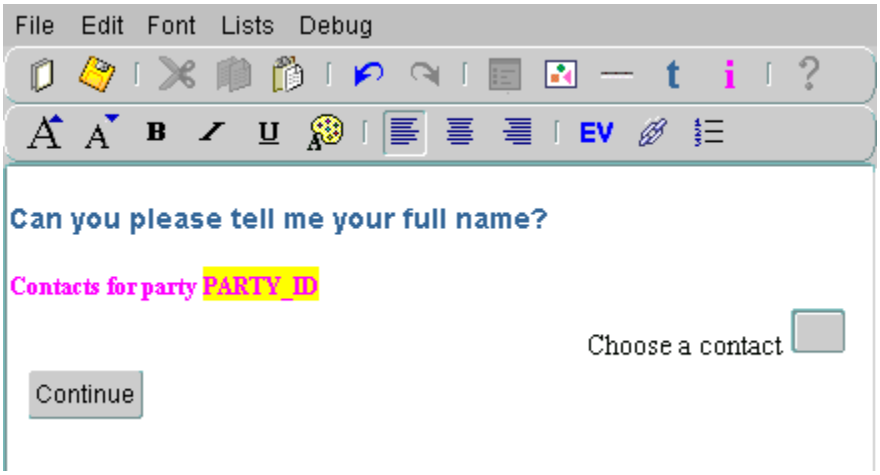
You can also include some complex functions, such as including embedded values in a panel, page 8-56.

An embedded value returns and displays information from a Script Author command in the panel at runtime. For example, if during the script session you obtain the customer's name, you can subsequently use a blackboard command as an embedded value command to display the customer's name in panel text, customizing the appearance of the script dynamically.

Using the panel layout editor, you can also create hypertext links, page 8-59, create ordered or unordered lists, page 8-55, and insert images into the text, page 8-57.

Example

The following image shows an example of the panel layout editor for a panel.



The example includes the following elements:

- Text, in dark blue, intended to be spoken by the script user at runtime ("Can you please tell me your full name?)
- Text, in magenta, to be displayed on the screen , but not necessarily to be spoken ("Contacts for party")
- Text with a yellow background ("PARTY_ID") - this shows an *embedded value*, whose actual value at runtime is evaluated and displayed through an Oracle Scripting command.

- Label text ("Choose a contact") immediately to the left of a button.
- A blank button that serves as a placeholder image for an answer, where in general at runtime the user either enters or selects one or more value.

  Note: In this case, the answer appears at runtime as a dropdown list of contacts for the current party, where the user needs to select one value.

  Note: The answer properties are defined in the Answers pane for the panel (not shown here).

- A Continue button

Panel Layout Editor Menu and Toolbars

The menu and top toolbar of the Panel Layout text Editor contain several options and buttons that fulfill many standard functions, such as New, Save, Cut, Copy, Paste, Undo, Redo.

Note: Keyboard shortcuts, such as Ctrl-X for Cut and Ctrl-Z for Undo, that are frequently used in other editing tools, do not apply in the Panel Layout Editor - you must use the menu options and toolbar buttons.

To the right of the standard function buttons in the top toolbar, the following buttons have specific Oracle Scripting functions:

- Modify Properties

  This is enabled only if you are editing Embedded Values or Hyperlinks

- Insert Image
- Insert Horizontal Line
- Set Spoken Text Font

  This is for text intended to be spoken by the script end user, as for example, a service agent.

- Set Instruction Text Font

  This is for text intended as extra information for the script end user, but not usually

to be spoken.

> **Note:** The two "Set Text" buttons represent a convention for intended use only. You can choose a special font or leave the text in standard font; Oracle Scripting does not enforce any intended usage at runtime.

The lower toolbar includes several industry-standard text formatting buttons, such as bold, italic, underline, left justify, right justify, and so on.

To the right of the formatting buttons, the following buttons have specific Oracle Scripting functions:

- Insert Embedded Value
- Insert Link
- Insert List

> **Note:** From the toolbar, the Insert List button enables you to insert a new ordered list.

> **Note:** From the Lists menu option, you can insert new ordered and unordered lists, as well as items into those lists.

Opening the Panel Layout Editor

You can open the panel layout editor in a number of ways.

The basic process is as follows:

1. On the canvas, right-click the panel > Edit Blob Properties.
2. Click Panel Layout.
3. Click Panel Layout Editor button.

You can also open the panel layout editor from the canvas as follows:

1. Click the Toggle Select Mode button in the Object and Branch toolbar.
2. Click the panel, then perform one of the following:

- Click the Panel Layout icon on the top toolbar.
- Enter Ctrl + Shift + E.
- From the menu, select Tools > Panel Layout Editor.

Entering and Formatting Panel Text

Note: Before you enter and format panel text in the Panel Layout Editor, Oracle recommends that you define panel question UI controls (answers), and values in the Label for Reporting field, if applicable.

General Text Handling

Type the panel text, then optionally, select text and then format it using one of the following methods:

- Select a format from the Font menu.
- To apply a preformatted text style, from the toolbar, click Set Spoken Text Font or Set Instruction Font.
- From the toolbar, apply font size, style, or foreground (text) color characteristics by clicking on the appropriate icon.
- To format paragraph-level alignment, from the toolbar, select Left Justify, Center Justify, or Right Justify, as appropriate.

Adding Lists and List Items

If you want to insert a bulleted or a cardinally ordered list, then do the following:

- Without selecting text first, place the cursor at the appropriate insertion point in the panel.
- From the Lists menu, select Insert Unordered List or Insert Ordered List, respectively.

  A bullet or the number 1. appears, indicating that you have started a list.

  (You can also start up a new ordered list by clicking the Insert List toolbar button.)
- Type or paste text into the list as appropriate. To add new items to the list, on your keyboard, press Return.
- To add items to lists, you can also select Insert Unordered List Item or Insert Ordered List Item from the Lists menu.

Add as many items and lists as you require.

Special Display Considerations in the Panel Layout Editor

Appearance of Answer Controls

Answer controls, images, and text can be interspersed in the panel layout editor, and formatted as desired. The panel layout editor displays placeholder question controls for any questions that are already defined for that panel.

Text entry question user interface control types (text fields, text areas, and password fields) and single checkbox controls appear the same in the panel layout editor as they will at runtime.

Radio buttons, drop-down lists, multi-select lists and checkbox groups display in the panel text editor showing only a single representative control (a single radio button, an empty list, or a single checkbox, respectively). This provides an indication of how the panel appears at runtime, and allows the script developer to format any question control labels or text.

The actual appearance of these controls can differ at runtime, since any of these question controls can be populated dynamically (for example, from a PL/SQL command, table lookup, or blackboard value command). Thus, these question control types are always rendered dynamically at runtime.

Editing Question Labels

If a label for reporting for a specific question is defined upon the initial definition of a question control using the Answer Entry Dialog window, then this value will also appear in the panel layout editor. However, the only time label content is generated in panel HTML is when the question control is initially saved.

If you subsequently edit the Label for Reporting field for a specific question, the change affects for all question-related and internal label identifiers but does not change the label on the panel layout.

So, if you modify the Label for Reporting field in the Answer Entry Dialog window after the data has first been saved,you must manually return to the panel layout and update the panel label accordingly.

The same is true if you subsequently remove a value initially entered into the Label for Reporting field; the label content will continue to appear in panel text until you manually remove it.

Inserting Embedded Values

Use this procedure to insert an embedded value into the panel.

Embedded values in panels act as placeholders for values that are dynamically entered into the panel text at runtime.

The basic process is as follows:

1. Type some placeholder text.
2. Highlight the placeholder text and then click Insert Embedded Value in the toolbar.

   The placeholder text gets a yellow background.
3. With the cursor in the text or with the embedded value text selected, from the toolbar, click Modify Properties.
4. In the Command window, define a Command that returns a value to be displayed in the script panel at runtime.

Example of Creating an Embedded Value

You must provide the known value of a blackboard key "First_name" to populate as the embedded value in a panel at runtime.

In the Panel Layout Editor, you perform the following:

- Select and highlight your placeholder text, click Insert Embedded Value**.**
- Click in the embedded value text, click Modify Properties**.**
- From the Command Type menu, select Blackboard Command, and enter First_name in the Command field. *The Name field does not influence the blackboard command.*

At runtime, as you enter the panel, the value of the blackboard key First_name is displayed in the panel.

Inserting Images

Use this procedure to insert an image into the panel.

The basic process is as follows:

1. Position the insertion point where you want to insert the picture.
2. In the toolbar, click Insert Image.
3. From your local file system or network, locate the .gif or .jpg file that contains the image you want to insert, click the file, then click Open.

   The image appears at the insertion point.

   > **Note:** The absolute path of the graphic is recorded in the panel HTML. This image will only appear correctly at runtime from workstations that can access the same path to render the image.

4. Optionally, make any layout changes necessitated by placement of the graphic into your panel layout.

Special HTML Considerations for Images in Panels

1. Imported images appear in-line with the text (not in an HTML table). To best control how images appear in a panel layout, you may want to modify the panel HTML to place the image in a table.

2. The panel HTML references the image in the absolute directory path from which it is inserted into the panel. This will only display properly at runtime to individuals who have that specific graphic image physically located in the same precise directory location on their computer's hard drive or network.

   To make an image visible to *all* users of a script, the image must be uploaded to a path accessible to the Oracle Applications Web server at runtime (for example, to $OA_MEDIA on the applications tier). Then, the HTML referencing the graphic image inserted into a script from a local or network location must be exported from the panel layout editor, and modified to reference the image in the appropriate relative path.

   The modified panel HTML code is then re-imported into the panel. When the script containing the modified HTML is deployed and executed, the image is available to all script end users at runtime.

Example of Image Related HTML Processing

1. Export the panel contents to an HTML File, page 8-59.

2. In the panel HTML code, locate the reference to the graphic image file.

   For example, if the file imported is a sample GIF image in the temp directory of your C drive, locate `<img src="file:///C:/Temp/sample.gif">`.

3. Replace the reference with the appropriate URL, adhering to project or company guidelines (such as specifying alternate text and image dimensions).

   For example, if you loaded the file onto the OA_MEDIA directory of the Oracle Applications server for Company.com, the revised image reference might appear as follows:

   ```
   <img src="http://server.company.com:8888/OA_MEDIA/sample.gif"
   width="200" height="50" alt="sample image">
   ```

4. Make any other required modifications to the panel layout HTML file, and save it in HTML format.

5. Import the file into the original panel, replacing the old panel layout contents with the modified HTML. For more details, see Importing HTML Files into the Panel Layout Editor, page 8-60.

Inserting Hypertext Links

Use this procedure to insert a hypertext link into the layout of a panel, and provide a URL for the link's destination at runtime

The basic process is as follows:

1. Highlight the appropriate text and then click Insert Link in the toolbar.

   The selected text is blue and underlined.

2. With the cursor in the text or with the text selected, click Modify Properties in the toolbar.

3. In the Hypertext Link window, in the URL field, type the full URL for the hyperlink. Include the protocol (e.g., HTTP://) in the URL field.

   For example, type http://www.oracle.com.

Saving Text

Performing a Standard Save

Choose File > Save from the menu.

Saving the Text as HTML

If you want to save the text to an HTML file, then choose File > Export from the menu, and specify a filename with an .html extension.

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

> **Note:** When importing panel HTML, only files with a full four-letter .html extension display, unless you explicitly change the File Type option in the Open dialog box to All Files (*.*).

Exporting Panel Text to HTML Files

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

> **Note:** Exporting the text to an HTML file does not save any changes made to the panel layout. If you want to save the modified panel as well as export it, you must explicitly save it in the panel by selecting File > Save.

The basic process is as follows:

1. Open the panel layout editor for a selected panel, page 8-54.
2. Select File > Export As.
3. Specify the file name you wish to designate to the exported HTML and the location on your local file system or network where you want the file to be saved, then click Save.

Importing HTML Files into the Panel Layout Editor

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

> **Note:** Importing an HTML file into the panel layout editor overwrites any content previously in the panel layout editor for the designated panel.

> **Note:** If question user interface controls were previously defined in the panel that have been removed from the panel HTML, then importing the modified HTML will delete those controls from the panel, and any corresponding data dictionary properties associated with the deleted question controls.

> **Note:** If importing panel HTML from one panel into another panel, no data dictionary properties from the first panel are included in your import. You must define data dictionary properties anew for any questions imported into a panel. If re-importing into a panel, any already existing data dictionary properties for questions previously defined in the panel are retained.

The basic process is as follows:

1. Open the panel layout editor for a selected panel, page 8-54.
2. Select File > Import.
3. Locate the file that you want to import, click the file, then click Open.

   > **Note:** Only files with a full four-letter .html extension display, unless you explicitly change the File Type option in the Open dialog box to All Files (*.*).

   The HTML appears in the panel layout editor.

4. Optionally, if you want data dictionary properties associated with any questions that are new to the panel, enter them as appropriate.

Guidelines

Neither the agent interface nor the Web interface of the Scripting Engine support the use of cascading style sheets (CSS). For executing scripts as surveys, any style sheet other than the default CSS specified in JTF properties will be ignored. The agent interface will ignore any style sheets referenced in panel layout HTML.

Customizing the Panel HTML

This section includes the following topics:

Background

The panel layout editor is *not intended to be used for complex layout.* For example, you cannot create HTML tables using this feature.

To create complex panel layouts, to use or modify HTML tables, or to change the value of automatically generated Continue buttons, it is expected that script developers will use full-function third-party HTML editors to modify panel HTML.

You can also create or edit panel HTML code by hand using a text editor.

Either of these approaches are considered customization. Customizing panel layout is also required for any panel intended to display GIF or JPG images within a panel.

Customizations Involve Importing and Exporting Panel HTML

You can import or export panel layout HTML from the panel layout editor.

If you want to customize panel layout, you have several options:

- Begin defining panel HTML by adding the relevant question UI controls to a panel in a graphical script, and optionally, entering and formatting text using the panel layout editor, and then export the panel HTML for subsequent modifications.
- Define all standard aspects of a panel by creating a script in the Script Wizard, and entering all panel aspects as appropriate. Subsequently, graph the script, and export panel content from the panel layout editor.

- External to Oracle Scripting, generate compliant HTML panel content, and then import the content into a graphical script using the panel text editor.

Regardless of your approach, all customized panel HTML must contain script-specific code, and not interfere with reserved words.

And regardless of the creation method of compliant panel HTML, the custom code must be imported into a graphical script using the panel layout editor, and deployed to the database in order to execute at runtime.

Guidelines

- Experienced HTML programmers may wish to define panel layouts in HTML instead of using graphical script tools or the Script Wizard, and then import them (panel by panel) into newly created empty panels in a graphical script.
  - Any panel properties not defined in panel HTML (for example, any data item visible in a question's data dictionary) must then be configured for each panel after import into a graphical script.
  - You must follow custom Oracle Scripting syntax and observe rules regarding panels and reserved words.
- There is no method to create or add JavaScript code from Script Author. To add your custom JavaScript code to an existing panel, you must export panel HTML, add the previously tested custom code, and reimport.
  - JavaScript is not supported in the agent interface.
  - JavaScript does not support global functions or functions external to a single panel in the Web interface. JavaScript restrictions are documented in detail in *Oracle Scripting Developer's Guide*.

Panel HTML Contains Script-Specific Code

Special syntax required for Oracle Scripting is enclosed in custom tags and should not be modified. Whether a script is created using the Script Wizard, or with Script Author graphical tools, any panel HTML must conform to certain rules and requirements specific to Oracle Scripting.

This includes:

- Custom tag names for question UI controls.
- An HTML table defining all questions, with the name property *IES_ControlManifest*.

  In panel HTML generated by Script Author, a single HTML table named IES_ControlManifest is created to hold all questions. By default, this table includes two columns. Columns and column properties can be modified, but this table must exist with the appropriate name property for panels to appear and function

appropriately at runtime.

- For panels with question UI types other than button, a line of code outside of the IES_ControlManifest table, but just prior to the close body tag. This code results in what is referred to as the Continue button. It is a control with the input value *Continue*, type property of *submit*, and the name property *IES_CONTINUE*.

  Sample HTML code for this control is as follows:

  ```
  <input value="Continue" type="submit" name="IES_CONTINUE">
  </body>
  ```

  If desired, the input value for this control can be modified. *No other properties must be changed for this control.*

- For graphical scripts only, you can also define a button control by defining a question UI control of type *Button*. If using this question control, it must be the only control in the panel. Panels with a button require you to specify one or more sets of answer choices (lookup values) in the data dictionary for that question.

  - For this control, the value property is the variable *pbLabel*; the name property is derived from the name of the question control in the Answer Entry Dialog window; and the type is *submit*.

  - Answer choices consist of a value that displays at runtime (the *display value*) and the value passed to the blackboard as the selected answer at runtime (the *value*).

  - When used to continue the script to the next panel, type the value Continue for both the display value and the value. This appears at runtime exactly like the automatically generated Continue button (which is generated for panels with other question control types).

  - This control type also supports the inclusion of more than one answer choice. Simply specify multiple sets of lookup values in the question data dictionary. This appears at runtime as multiple buttons, horizontally displayed. Clicking any answer choice button registers the response for that panel at runtime.

  - Because you can include one value or multiple values, this control type is referred to in HTML code as a *pushbutton group* (pbgroup). Except when specifying HTML code, product documentation refers to this control as a button (or a submit button).

  - In contrast to auto-generated Continue buttons, the HTML code for this control falls *within* the IES_ControlManifest table.

  - Because it can support multiple answer choices, the HTML code for this control (when exported from the panel layout editor) does not specify display values.

The code contains a variable, *pbLabel*, which is dynamically populated in the panel at runtime. Like the radio button, drop-down, checkbox group, and multi-select list box question UI controls, this value is provided to the Scripting Engine from the question data dictionary at runtime.

- Sample HTML code for this control is as follows:

```
<pbgroup>
<input value="pbLabel" name="question name, as entered into the
Name field in the Answer Entry Dialog window" type="submit">
</pbgroup>
...
```

Syntax for all question controls is documented in detail in *Oracle Scripting Developer's Guide*.

What Is Not Supported in Panel Layout HTML

Generally, panel HTML must conform to the HTML 3.2 specification, and should be editable in any HTML 3.2 compatible editor.

> **Note:** Not all tags accessible to HTML 3.2 are supported in Oracle Scripting, and some support is limited.

The following list includes some limitations of Oracle Scripting. This list is by no means all-inclusive.

- Oracle Scripting does not support the embedding of Applets within panels.
- Oracle Scripting does not support the use of cascading style sheets (CSS) within panel HTML.
- Oracle Scripting does not support the SCRIPT tag.
  - Some use of this tag will function but is not guaranteed. *Use this tag at your own risk.*
  - The use of JavaScript in an HTML page outside of the start and end body tags is not supported.
  - JavaScript is not supported in the agent interface.
- Global variables defined in panel HTML do not persist from panel to panel.
- Panels containing a button question UI control are not allowed to contain any other question UI controls in the panel.

- No two controls in the same panel HTML may have the same name.

You must understand how Oracle Scripting functions in order to understand these limitations. Some facts supporting the reasons for the above restrictions are included below.

- Scripts are designed to execute in at least two Scripting Engine interfaces (one a Java applet, the other is interpreted by a Web browser). The lowest common denominator (the agent interface) is used to enforce uniformity of behavior.

- The agent interface uses older implementations of Java SWING and other classes to interpret HTML. Until all of Oracle Applications uptakes Java Virtual Machines or Java Runtime Engines that interpret code from more recent HTML specifications, the limit remains HTML 3.2.

- Because the Scripting Engine agent interface is a Java applet, panel HTML is interpreted by SWING classes, not by a modern Web browser. This explains why JavaScript is not supported by the agent interface.

- In contrast, panel HTML for scripts executed in the Web interface are interpreted by the Web browser. Thus, you must use an Oracle Applications certified Web browser. Note that HTML is rendered differently in different supported browsers, which can result in viewing differences based upon which compliant Web browser is used to execute a script.

- When scripts are executed in the Web interface, JavaServer Pages (JSP) execute, and panel HTML is passed to the primary JSP page. Panel content is passed by the Scripting Engine from the panel to the executable JSP, resulting in the stripping out of content prior to and following the open and close body tags.

    - This results in limitations with applets, CSS, and the script tag.

    - Because the Scripting Engine considers only panel HTML content between the open and close body HTML tags, and other information is discarded at runtime, global variables defined in panel HTML do not persist from panel to panel.

- The JSP page controls the script's user interface and appearance at runtime, based in part on:

    - Whether the script is hosted in a self-service Web application (hosted) or not (is a standalone script).

    - Options selected in the survey campaign requirements (including the appearance of optional header and footer sections, and specification of error and final pages or page redirects.

Working with Groups

This section consists of the following topics

Group Properties and Attributes

A group object is a container for other objects and branches, and always contains one subgraph, also known as a child graph. That subgraph may contain any type of object, including other groups.

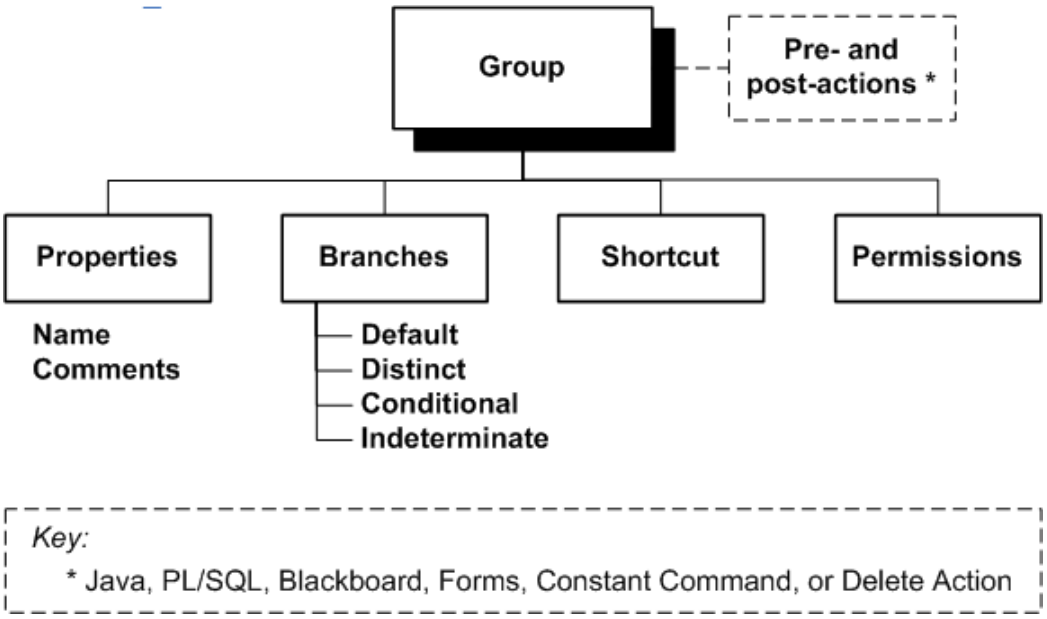
Groups can be nested as many levels as desired; the only limit is practicality. Each group must meet certain requirements (see Minimum Requirements for Any Graph, page 2-27).

Groups are used to:

- Logically group a section of the script flow - typically, to contain a single function or process
- Serve as a container for a shortcut, so that at runtime a shortcut button can appear in the GUI that will "jump" the user to that group
- Restrict the access of certain or all users to a section of the script at runtime

In addition, when a saved script is imported into a new script, it is imported as a group.

The main group properties and attributes appear in the Group Properties window, as shown in the diagram following.



These panel properties are accessible in the Script Author by right-clicking a canvas group and selecting File > Edit Blob Properties, or if you have enabled auto-popup when creating a new panel, group, or block.

For more information about actions and branches, see:

- Defining Actions, page 8-88
- Working with Branches, page 8-82

Notes on Group Properties and Attributes

Name

Identifies the group in the Script Author and the database.

Shortcut

If you want the group to be the target or destination of a "jump" from another location of the script, specify a shortcut name. You can only have one shortcut name associated with the group.

For more details about shortcuts, see Defining Shortcuts, page 8-68.

Permissions

You can allow or limit access to the group by specifying permissions, to the level of individual users.

Inserting Groups

For general information on inserting groups, see Inserting Objects and Branches, page

8-9.

For details of group properties, see Group Properties and Attributes, page 8-66.

Using Subgraph Information

The subgraph of the group, which is executed before the group, may contain panels, blocks, and other groups. You can collect information from the subgraph objects to be used in the containing group.

For more information, see Using Subgraph Information in Blocks and Groups, page 8-81.

Defining Shortcuts

A shortcut is a property of a group in Script Author. The shortcut exposes the functionality of the group and its contents to the Scripting Engine, enabling that group to be the target or destination of a "jump" from another location of the script.

This jump occurs when the shortcut is invoked using the Oracle Scripting jumpToShortcut API. This API is associated with the script using a Java command, which can be called as an action, pre-action or post-action anywhere in a script.

The shortcut property of a group is also used by the indeterminate branch. This branch type requires an expression to be defined. This expression is a Java method which, when evaluated, provides a shortcut name as its return value (based on the conditions tested in the method), invoking the shortcut.

Regardless of whether it is called from an action or an indeterminate branch, when the shortcut is invoked, the flow of the script at runtime jumps from the existing object being processed to the group containing the shortcut name that is returned by the command.

Script flow is thus changed, and flow of the script at runtime resumes from the target group. If the destination group contains a panel, that panel is the next to display to the script end user. Otherwise, processing continues of all script objects, and the next panel placed in the revised flow displays at runtime.

Four typical uses of a shortcut are as follows:

- To associate a target for a shortcut button in the shortcut button area (displays for the agent interface only).
- To enable the Disconnect button (displays for the agent interface only).
- To associate a target for a group containing specific functionality which must be accessed after meeting a specified condition in a script (for either runtime interface).
- To redirect the flow of a script to a new path (starting with the group that contains the shortcut).

Use this procedure to define a shortcut, specifying that group as the target of a jump.

Prerequisites

Insert a group.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a group.
3. In the Group tree, select Shortcut.
4. In the Shortcut pane, in the Shortcut field, type the name of the shortcut.

   For example, to define a group as the target after clicking Disconnect in the agent interface, type WrapUpShortcut.
5. Click Apply to save your work and continue, or click OK to save your work and exit the Properties window for the group.

Guidelines

- If you want to jump to the shortcut in runtime from a shortcut button, define the shortcut button.
- If you want to jump to the shortcut in runtime based on values or other criteria, define that criteria and associate the criteria with one or more commands in the script.
- If you created this group to enable the Disconnect button in the agent interface, you must ensure the subgraph created by the group is appropriately terminated, with, as a minimum, a default branch from the start node to a termination node.

Importing Saved Scripts as Groups

When you import a script, the entire script - the root graph, and all subgraphs - is placed into your current script, as a group, whose name is the script name of the imported script.

This procedure is described in the task called Importing Scripts, page 8-17.

Working with Blocks

Using blocks, you can integrate data sources into your script. Block types include insert, query, update, and API blocks.

Using insert, query, and update blocks, you can add to, select from, or update database tables from a script, respectively. You provide the necessary information as you define

the block properties; at runtime, Oracle Scripting generates and executes the appropriate SQL command.

Using an API block, you can add any Script Author command (Java, PL/SQL, blackboard, forms, constant, or a delete action) to your script.

When you create a block, you automatically create a subgraph, also known as the *child graph*. You can either leave the subgraph as a minimum graph (one start node, one termination node), or you can add in panels, blocks, and groups.

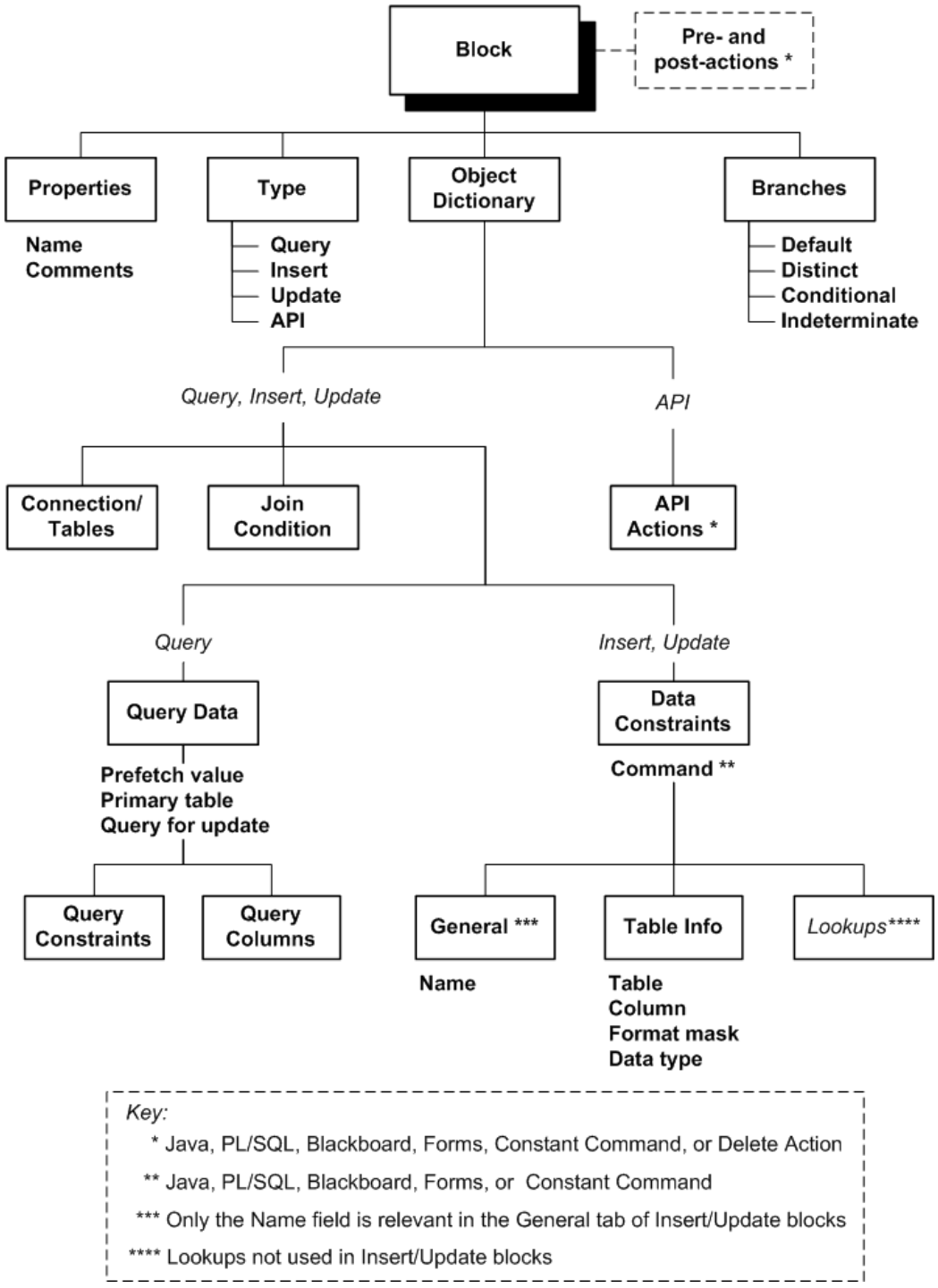
The objects in the subgraph are evaluated before the main command of the parent block. This enables you to solicit data that may be passed to the parent block command. You can validate the extra data and even prevent the execution of the parent block command.

This section consists of the following topics:

Block Properties and Attributes

The key block property is Type, and the four allowed values are Query, Insert, Update, and API. The block type value controls which other properties are relevant for the block type; these extra context-sensitive properties are defined in the Object Dictionary associated with the type of block chosen.

The main block properties appear in the Block Properties window, as shown in the diagram that follows.



For more information about actions and branches, see:

- Defining Actions, page 8-88
- Working with Branches, page 8-82

For each block type, the Object Dictionary contains information relevant to that block type. The information categories are reflected in the tabs and subtabs that appear as you create and edit the block, as follows:

- *Query block* categories:
  - Connection/Tables, page 8-73
  - Join Condition, page 8-73
  - Query Data (with sub-categories Query Constraints and Query Columns), page 8-74
- *Insert and Update block* categories:
  - Connection/Tables, page 8-73
  - Join Condition, page 8-73
  - Data Constraints (with two sub-categories General and Table info), page 8-75
- The main components of an *API block* are Oracle Scripting commands; you can define one or more commands in the API Actions window, page 8-76.

Notes on General Block Properties and Attributes

Name

Identifies the block.

Type

Identifies the block type: Query, Insert, Update, API.

Notes on Block Object Dictionary Properties

This section consists of the following topics:

- Connection/Tables tab, page 8-73
- Join Condition tab, page 8-73
- Query Data tab, page 8-74
- Data Constraints tab, page 8-75
- API Actions tab, page 8-76

Connection/Tables tab

Specifies the connection to a database and one or more tables that exist at that database. The connection is either the login connection or a previously-defined remote connection to the target database.

For more information about connections, see Working with Database Connections, page 8-86.

For query blocks, you can specify one or more tables.

For insert and update blocks, you generally specify one table.

> **Note:** List master tables first and then tables related by foreign key.

Join Condition tab

Used to specify primary key and foreign key data and relationships between a master and a detail table when you define two or more tables in the Connection/Tables tab, page 8-73.

When you add a join condition in this tab, three subtabs appear, into which you must enter the appropriate data.

To illustrate the process, assume the following example:

- *Master table is Customers, primary key is Cust_Id.*
- *Detail table is Orders, primary key is Order_Id, foreign key Cust_Ref references Cust_Id in the Customers table.*

For each master-detail table relationship, you must enter data as shown in the following table *(the italicized column shows the values for the example tables)*:

Subtab	Values to Enter	*Example Values*
Master PK Table	Master table name, and its primary key	*Customers, Cust_Id*
Detail PK Table	Detail table name, and its primary key	*Orders, Order_Id*
Detail FK Table	Foreign key from the detail table (that references the master table primary key)	*Cust_Ref*

For multi-column primary and foreign keys, enter the column names on separate lines in the unnamed text area of each subtab.

> **Note:** Where there is only one table defined in the Connections/Tables

tab, you can also specify Master PK Table information in the Join Condition tab, but this is optional.

Query Data tab (for Query blocks only)

Prefetch value

Specify the number of rows to be returned for each database query, if you want to improve network performance. The default setting for -1 can be retained.

The row prefetch value is a feature of Oracle JDBC drivers; the default setting returns 10 rows at a time.

Primary table

Used to enable the processing of certain Delete Action commands.

In general, a Delete Action command deletes the row corresponding to the current cursor row from the data retrieved during the last database query. The Primary table field can be used when the last database query was for data from more than one table.

If all the following conditions are true:

- The cursor data was retrieved from more than one table
- The tables are connected by primary-foreign key relationships
- The top table (TT) of the table hierarchy has a Cascade Delete constraint defined

then, in order to allow the Delete Action to delete rows from more than one table, you must set Primary table to the top table of the table hierarchy.

> **Note:** The deletion that occurs during the execution of the Delete Action command does not display a popup window warning you that the record will be deleted. If you have to delete records in scripts, Oracle recommends that you use PL/SQL business API's for processing deletes, in PL/SQL commands or in API blocks.

Query for update

Specifies if this block is to be used to retrieve data, which will be subsequently updated in an Update block.

Query Constraints subtab

- Specifies conditions to restrict the rows to be returned from the query.

  Examples:

  1. To limit the rows from a Customers table to those where the Acct_Num column is 12345, enter the following condition:

```
Customers.Acct_Num = 12345
```

2. To restrict the rows from a Customers to those where the Cust_Dept column is SALES, enter the following condition:

```
Customers.Cust_Dept = 'SALES'
```

3. To restrict the rows from a Customers table to those where the phone number matches a blackboard value (EnteredPhone) retrieved and stored earlier in the script, enter the following condition:

```
Customers.PhoneNbr = :EnteredPhone
```

> **Note:** Make sure that, between ":" and "EnteredPhone", there is no space.

4. If your last query retrieved a single row from the table Cust_Info which included the Phone_Number column, enter the following condition:

```
Customers.PhoneNbr = :Cust_Info.Phone_Number
```

> **Note:** In these examples, the mechanism by which you specify blackboard values is through bind variables. If bind variables occur in more complex code such as within DECODE statements, make sure each bind variable is followed by a space. For example, enter
>
> DECODE (:ValType ,'X','Do Nothing',:Valtype) and not
>
> DECODE (:ValType,'X','Do Nothing',:Valtype)

> **Note:** You can insert logical operators (such as, NOT, AND, and OR) either to combine two conditions within one constraint or at the end of a 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.

Query Columns subtab

Specifies the data columns to be returned from the query.

Data Constraints tab (for Insert and Update blocks only)

Command

The command returns the new data value for the insert or update command.

You define the target column for the new data in the Table info subtab.

General subtab

Name

Identifies the data constraint.

After the data constraint command is executed, the value is stored in the blackboard under the name of the data constraint.

Table info subtab

Specifies the target destination of the data value returned by the Data Constraints command.

For insert blocks, this identifies the table and column to insert into. For update blocks, this identifies the table and column whose data will be updated.

API Actions (for API blocks only)

Specifies the command that executes when the block is processed at runtime.

Inserting Blocks

For general information on inserting blocks, see Inserting Objects and Branches, page 8-9.

After creating a block on the canvas, you can then define the function of the block by selecting the block type and specifying further block properties.

For details of block properties, see Block Properties and Attributes, page 8-70.

You can also navigate down to the child graph of the block and create objects there.

Defining Query Blocks

Use a query block to obtain information from one or more database tables. This information can be used for processing within a script, or as a prerequisite to updating a database table through an update block.

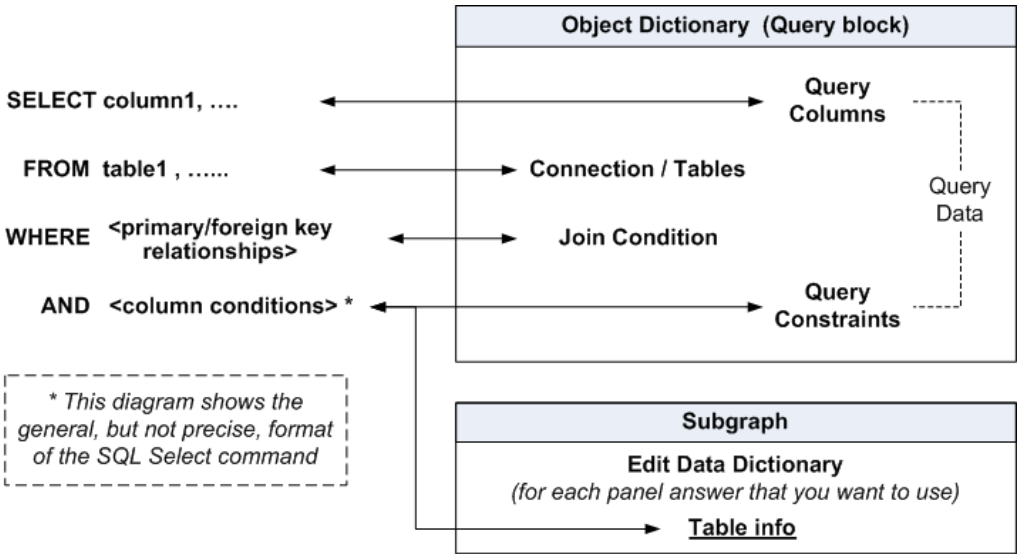
Using Subgraph Information

The subgraph of the query block, which is executed before the block query, may contain one or more panels. Panels within a block allow the script to collect answers to be used as constraints for the block query.

For more information, see Using Subgraph Information in Blocks and Groups, page 8-81.

Defining Query Block Properties and Attributes

Oracle Scripting generates a SQL Select statement from the properties that you define for the block. An overview of the Select statement elements and where you must define the corresponding properties appears in the following diagram:



The setting up of the query block's Object Dictionary properties for the SQL Select statement can be summarized as follows:

- Specify the columns to query in the Query Columns subtab of the Query Data tab, page 8-74.
- Specify the table information in the Connection/Tables tab, page 8-73.
- Specify any primary and foreign key relationship for multi-table queries in the Join Condition tab, page 8-73.
- Specify other constraints in the Query Constraints subtab of the Query Data tab, page 8-74.

> **Note:** You can also specify dynamic constraints that depend on answers retrieved at runtime, by defining Table Info properties for answers in panels of the block subgraph. For more information, see Answer Properties in the Edit Data Dictionary Window, page 8-36.

The full set of query block properties and how to define them appears among the Block Properties and Attributes. Use that topic and the preceding diagram to enter the properties for the query block, after you have inserted the block in the canvas.

Scripting Cursor Usage

Records returned by a query are stored in the scripting cursor. This set of information can be manipulated by Oracle Scripting cursor APIs, or used as answer choices to questions in a panel.

The data in the first record in the cursor can also be displayed in panel text: you can display the individual column values that are part of the record in panel text as

embedded values.

If you need to display a record other than the first record in panel text, you can use cursor APIs to manipulate the record pointed to by the cursor.

Information returned by a query is stored and accessed by its table and column name, in format TABLE.COLUMN.

Information returned by a query block is only stored in the cursor until a subsequent query is executed, or until the script session is completed.

Retaining Retrieved Information

To retain information returned by a query, you can use Script Author commands.

- For example, you can use a blackboard command as a parameter to a Java command to store the information in the scripting blackboard for the full duration of the script transaction, regardless of whether one or more additional queries are performed.

  > **Note:** Make sure you use a different blackboard key (and not TABLE.COLUMN) as it could be overwritten by a different query or if the cursor record is traversed.

- You can also use a PL/SQL command to call a PL/SQL procedure defining a global or local variable. This also keeps the value accessible to the Scripting Engine only for the duration of the script transaction.

- If necessary, you can use a PL/SQL command to write the values to a custom table. This should be used sparingly, however, as it can impact performance of the script. Such functions are generally recommended to be performed as a post-action to the script, so the impact to script end users is minimized.

Defining Insert and Update Blocks

> **Note:** Insert and update blocks are one way that you can use in Oracle Scripting to interact with database tables. Because of their greater flexibility, Oracle recommends that you call business API's to insert and update data, in PL/SQL commands or in API blocks.

Using an insert or update block, you can gather information and either write to or update records in one or more database tables specified within the data dictionary of the block.

Insert blocks are independent of any other coding.

The general format of the Insert statement generated by Oracle Scripting is:

```
Insert into TableX (col1, col2, ...,coln) values (val1, val2, ...,valn);
```

You specify the target columns and the values for those columns within the insert block and its related subgraph.

Update blocks are defined to update data that must have been retrieved by a previous query block.

The general format of the Update statement generated by Oracle Scripting is:

```
Update TableY set col1=val1, ....,coln=valn where <data constraints>;
```

The rows to be updated, or their ROWIDs, must exist in the cursor. The update block and its subgraph define both the columns to be updated and their replacement values.

> **Note:** Oracle recommends including code in your script prior to an update block to validate that the prerequisite query returns at least one valid row.

Using Subgraph Information

The subgraph of the block, which is executed before the block insert or update, may contain one or more panels. Panels within a block allow the script to collect answers to be used in the block insert or update.

For more information, see Using Subgraph Information in Blocks and Groups, page 8-81.

Defining Properties for the Insert or Update Block

Oracle Scripting generates one or more SQL Insert or Update commands from the properties that you define for the block.

You define the table name for the insert or update in the Connection/Tables tab of the Object Dictionary for the block, page 8-73.

Generally, you specify a single table name in the Connections/Tables tab, and you do not need to enter any properties in the Join Condition tab.

For both inserts and updates, you must define the target columns and the new data values for those columns.

This information can be provided from two places:

1. For columns whose values are provided by the script end user at runtime, as an answer to a panel question, define the target column in the Table info area of the panel answer properties, page 8-36, in the subgraph associated with insert or update block.

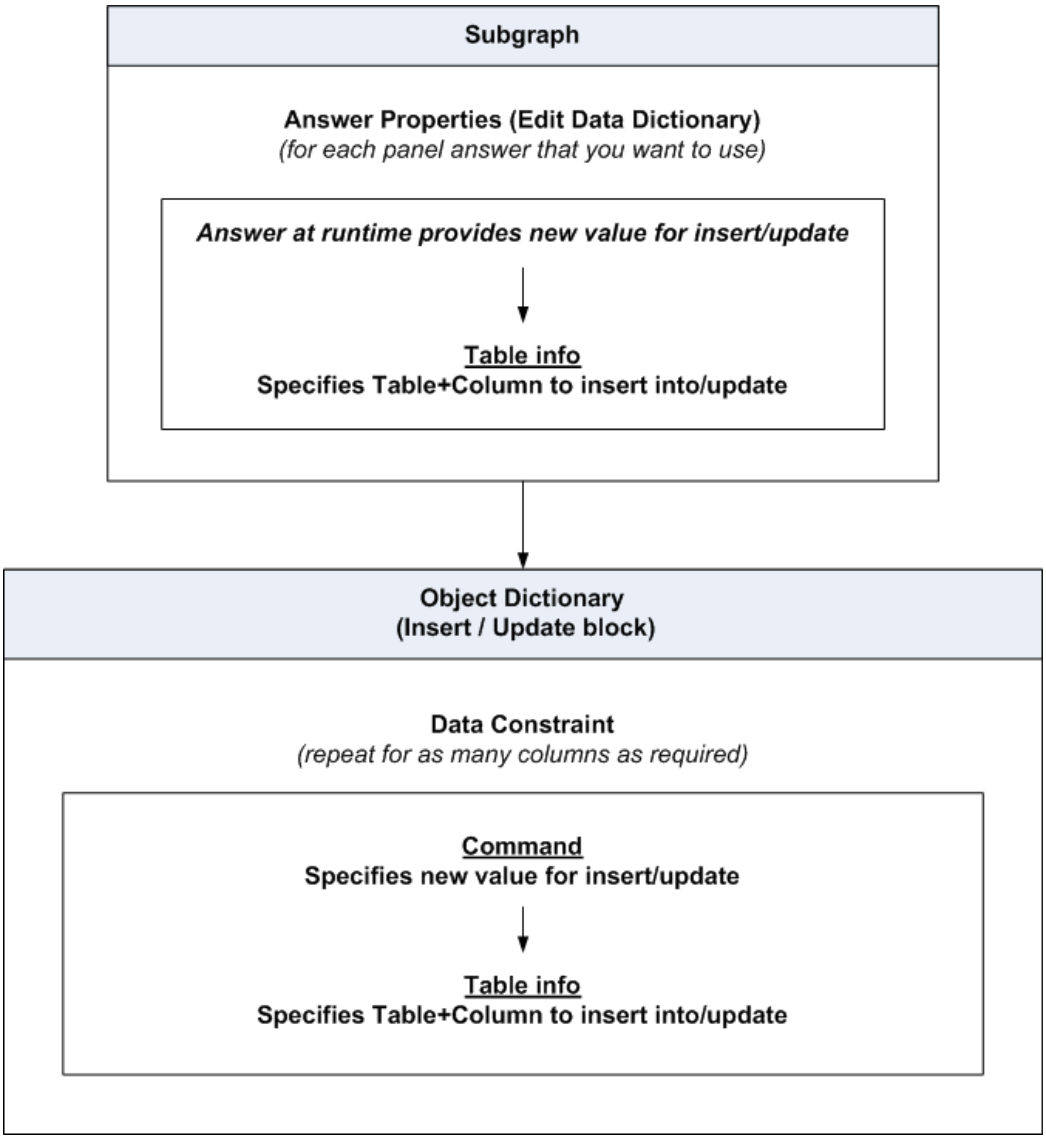
2. For other columns, you must specify the following information in the Data Constraints tab of the Object Dictionary of the insert or update block, page 8-75:

   - The target column in the Table info tab

   - The Command that returns the new value for the target column. For more

details, see Commands to Provide Column Values, page 8-80.

These requirements are summarized in the following diagram.



The full set of insert and update block properties and how to define them appears among the Block Properties and Attributes, page 8-70. Use that topic and the preceding diagram to enter the properties for the insert or update block, after you have inserted the block in the canvas.

Commands to Provide Column Values

You must use the Data Constraints tab, page 8-75 to define commands to provide column values for the columns to insert or update, in one of the following ways:

- You can define a blackboard command to obtain information from the blackboard to use in the insert or update.
- You can define a PL/SQL command to obtain information from a database table or view to use as a parameter for the insert or update.
- You can define a Java command to obtain the required data to use in the insert or update.
- You can define a Forms command to obtain the required data from an open form to use in the insert or update . Not all Forms-based applications allow Oracle Scripting to obtain information from the form.

For details, see Defining Commands, page 8-70.

Defining API Blocks

API blocks provide the flexibility of using any type of Oracle Scripting command - Java, PL/SQL, Blackboard, Forms, Constant - or Delete Action, at any point in a script flow.

After you have inserted a block into the canvas, the only block properties required for an API block are the block type, which must be API Block.

Define one or more Commands in the API Actions tab, page 8-76.

For general information about Commands, see Defining Commands, page 8-70.

Using Subgraph Information in Blocks and Groups

The subgraph of any block or group, which is executed before the block or group, may contain one or more panels. These subgraph panels allow the script to collect answers to be used during the execution of the parent block or group.

Use the buttons in the Navigation Toolbar to move between the block and its subgraph.

For query blocks, for example, you can include a panel in the subgraph that requests an account number. You can specify that the account number entered at run time should act as a constraint on the parent block's query, in addition to any constraints explicitly defined for the query block.

For insert and update blocks, for example, you can include a panel in the subgraph that requests a product name. You can specify the product name entered at run time should be either inserted into a database column or that it should update an existing database column.

For groups, as well as for all blocks, you can store panel answers entered at run time into the blackboard, and make them available in the parent block or group.

Especially for insert blocks, you can test that all required data has been provided within the subgraph, using a conditional branch. For example, you can define a Java method as

the expression for the conditional branch which tests for a specific condition (the presence of all required information). The conditional branch, created at the end of the flow for objects within the block, leads to the termination node of the block, exiting the block flow (and executing the insert action) only when all required information is accessible. A default branch (with a lower branch order) can direct flow to panels or other objects which request the information until all required data is not yet available.

For groups, as well as for all blocks, you can store panel answers entered at run time into the blackboard, and make them available in the parent block or group.

For more details about the use of panel answers in block subgraphs, see the information about Table info columns in the topic Answer Properties in the Edit Data Dictionary window, page 8-36.

Working with Branches

Branches are generally properties of the configurable object (panel, block, or group) from which they start.

To view and edit branches, first access the Properties window of the start object.

For general information about working with branches, see the following topics:

- Inserting Objects and Branches, page 8-9
- Editing Objects and Branches, page 8-10
- Deleting Objects and Branches, page 8-11

This section, which contains more specific branch-related tasks, consists of the following topics:

- Branch Properties, page 8-82
- Reordering Branches, page 8-84
- Adding Corners To and Removing Corners From Branches, page 8-85
- Changing Destinations of Branches, page 8-85
- Considerations for Indeterminate Branches, page 8-85

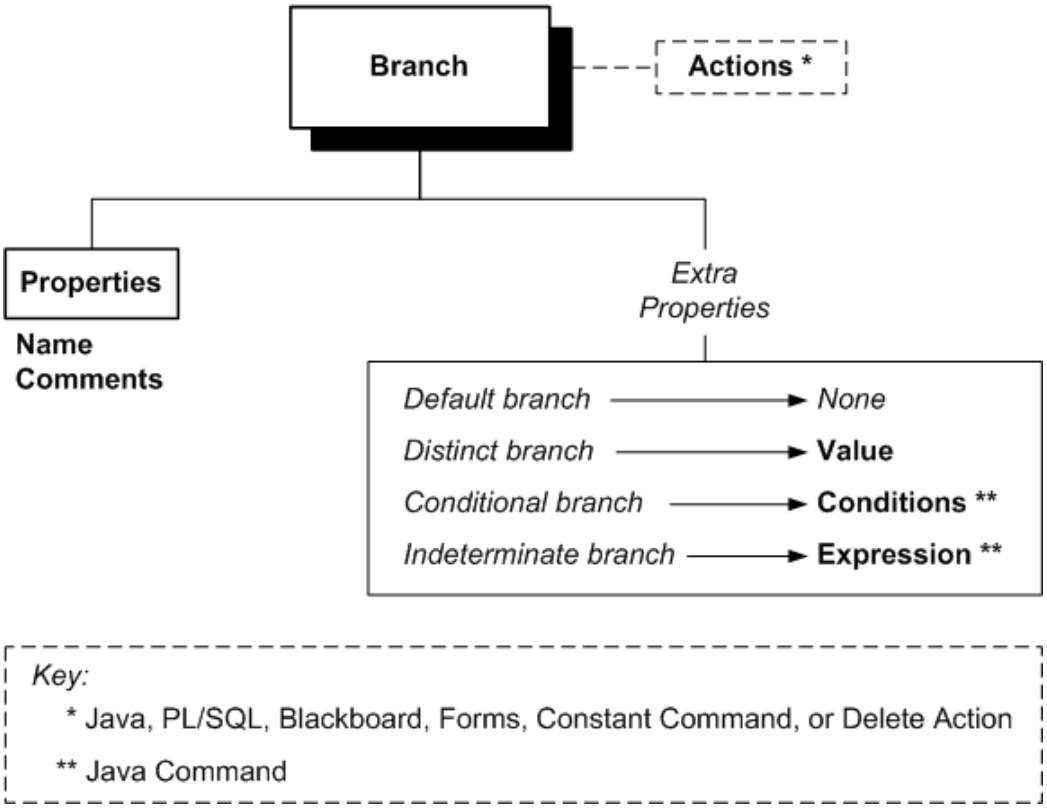
Branch Properties

You can attach branches to panels, blocks, nodes, and groups.

> **Note:** Special considerations apply to start nodes - *each start node must have a single default branch* - and termination nodes - *by their very nature,*

they can only be the targets of branches.

Within each object, the main properties of branches that you can define appear in the diagram following:



For details of the actions, see Defining Actions, page 8-88.

Guidelines for Defining Branches and Branch Properties

1. Each default, distinct, and conditional branch connects two configurable objects or nodes.

2. From a start node, you can create only a single default branch.

3. A *distinct branch* can only start from a panel.

   The panel must have one answer that is designated as the Default for Distinct Branching. For that answer, you must provide a set of lookup values.

   For each distinct branch, you must select a **Value** from one of the panel answer lookup values. You do not have to have a distinct branch for each lookup value.

4. For a *conditional branch*, you must specify, in the **Conditions** field, the name of the Java code that provides the Boolean expression to be evaluated at runtime; the

conditional branch is chosen if the code returns the value True.

5. For an *indeterminate branch*, you must specify , in the **Expression** field, the name of the Java code that evaluates the destination at runtime. For more details, see Considerations for Indeterminate Branches, page 8-85.

> **Note:** When you add a condition or an expression for a conditional or indeterminate branch respectively, you must select Java Command for the command type in the Command Properties window.

> **Note:** For conditional and indeterminate branches, during a script syntax check, there is no validation to determine whether the custom code referenced by the Java command exists.
>
> However, the referenced custom code must be available in runtime in order for the script to execute. Thus, any custom code must be appropriately uploaded or deployed to the server and accessible by the Scripting Engine so the code can be referenced by the script at runtime.
>
> For more information, contact your systems administrator or refer to the *Oracle Scripting Implementation Guide*.

Reordering Branches

Below the start node, each object placed on the Script Author canvas in a graphical script is typically connected above and below by appropriate branches, ending with the termination node. Outgoing branches (branches drawn *from* any object) are displayed in the Branch pane that results by clicking the Branch attribute under any object's Property window.

For each object to which they are attached, the branches are evaluated at runtime in the order that they appear in an object's Branches pane, where you can reorder them.

> **Note:** For a panel with multiple branch types, it is important to ensure that the Default branch is listed last so that other branch types will be evaluated.

To reorder a branch, select it in the Branches pane, and click either Move Up or Move Down as appropriate.

Adding Corners To and Removing Corners From Branches

> **Note:** In order to perform these operations, if the Toggle Select Mode button on the Object and Branch toolbar displays a raised image, you must click the Toggle Select Mode button.

When you create a branch in the canvas by dragging directly from a source to a destination object, the branch appears as a straight line on the canvas. You can create corners in the branch as you create it by releasing the mouse at one or more points before you reach the destination object.

You can add a corner to an existing branch by clicking on any point of the branch and dragging away.

To remove a corner from a branch, click on the corner - a cross-hairs icon appears as you hover directly over it - then either press the Delete key on your keyboard, select Edit > Delete from the menu, or click the Cut toolbar icon.

Changing Destinations of Branches

> **Note:** This procedure requires you to delete the destination object connected to the branch.
>
> If you want to retain that object for use elsewhere, then first select the object, use the Copy command to copy the object, and use the Paste command to place the copy of the object on the canvas.

The basic process of changing the destination of a branch is as follows:

1. Click the Toggle Select Mode button.
2. Select the current destination object of the branch.
3. Select Edit > Delete.
4. Select or create the new destination object.
5. Drag the destination object to the branch.
6. When the branch turns red, release the destination object.

Considerations for Indeterminate Branches

1. Use an indeterminate branch when the destination of the script is uncertain until information is gathered in a script session and evaluated.

Such information can be:

- End user responses to panel questions.
- Relative logical conditions such as the time of day or day of the week.
- Information received into the scripting cursor or the scripting blackboard from a PL/SQL package or a PL/SQL command executed during the session.
- Any other logic specified in a Java method referenced by a Script Author Java command (including custom code or best practice Java commands).

  You must include all the relevant factors in Java code, the return value of which dynamically determines the target destination for the script flow when evaluated at runtime.

2. The return value of the Java code must be one of the following:
   - A *sibling* object, that is, a panel, group or block, on the same graph as the indeterminate branch.
   - The shortcut name of a group at any level of the script.
3. To reflect the fact that the destination is unknown until runtime, the indeterminate branch is drawn on the Script Author canvas from the source object to an open point on the canvas; it does not connect to another script object.
4. The indeterminate branch is the only branch type that does not connect two objects. Use of this branch type allows that graph on the canvas to break two syntax requirements: (1) the requirement for every object that will be evaluated at runtime to be connected with branches, and (2) the requirement for the graph containing the indeterminate branch to include a termination node.
5. A termination node is still required if the destination object (the destination object to which flow of the script will progress at runtime) is on the same graph as the indeterminate branch, because the Scripting Engine needs to be apprised of when to end processing for that graph.

Working with Database Connections

The SQL command generated by Oracle Scripting for a query, insert, or update block is defined to execute at runtime either at the same database instance that you log into when starting the Oracle Applications session to use the Script Author applet - the login connection - or on another database - the remote connection.

By default, the initial connection property is set up for the SQL command to use the login connection.

List for Remote Connections

If you want any generated block SQL command to execute remotely, you must first define the remote connection or connections in a Connection List for the script. You can then select the appropriate remote connection for each block as required.

Use the Script Author Connection List (Script menu > Connection List) to define tables or views to which you want to connect to perform SQL actions (or any other database function).

Selecting a Connection for a Block

For any query, insert, or update block, you can then select the appropriate connection in the Connection/Tables tab.

The connection options are:

- Use Login Connection: The connection to the database instance which was established when your Oracle Applications user logged into Oracle Applications.
- You do not need to define the login connection using the Connection List.
- Reuse an existing connection: When you select this option, the connection drop-down list is enabled, in which all connections defined using the Connection List appear.

> **Note:** For remote connections, data changed in one remote connection session cannot be seen in another remote connection session, as the commits are not done until the transaction is finished. The login connection does not have this issue as the same "db session" is shared throughout the script transaction.

Defining Remote Database Connections

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

The basic process is as follows:

1. From the top menu, select Script > Connection List.
2. In the Connection tab, click Add, then enter the following data:
   - Connection Name
   - Host Name *(include the domain if appropriate)*
   - Port Number
   - SID

- User Name.
- Password

3. You can optionally test the connection in the Test tab

> **Note:** When you import a script into another script, the script connections from the imported script are added to the connection list of the target script.

Defining Actions

Actions are commands that can be added to a script and to any of the configurable script objects and branches. The actions can be one or more of the following:

- Command types: Java, PL/SQL, Blackboard, Forms, or Constant
- Delete Action

Pre-Actions and Post-Actions

You can define pre- and post-actions for the following Oracle Scripting elements:

- Script
- Panel
- Group
- Block

Script pre-actions execute in the Scripting Engine interfaces before the first script panel appears; script post-actions execute after the last panel is executed in the script.

Each panel, group, and block can have both pre-actions and post-actions, which execute correspondingly before and after the runtime evaluation of the specific panel, group, or block.

The basic process of adding a pre- and post-action is as follows:

1. Access the appropriate Properties window for the script, panel, group, or block.
2. Expand Actions, then select Pre Actions or Post Actions
3. Click Add, then enter details in the Command window.

   For details, see Defining Commands, page 8-89.

Actions

You can also explicitly attach actions to:

- Branches
- API Blocks

Actions attached to a branch execute at runtime as soon as the script flow goes through the branch.

The basic process of adding an action to a branch is as follows:

1. Access the Branch Properties window.
2. Click Actions, click Add, then enter details in the Command window.

   For details, see Defining Commands, page 8-89.

   > **Note:** You can add an API block to reference a Script Author command instead of associating an action with a branch or a pre- or post-action with a panel, group, or block. This provides the added advantage to script developers of a clear visual indicator that an action is referenced.

The basic process of adding an action to API Block is as follows:

1. Access the Block Properties window.
2. Click Object Dictionary, click Add, then enter details in the Command window.

   For details, see Defining Commands, page 8-89.

Defining Commands

You can perform the following tasks:

- Defining Java Commands, page 8-90
- Defining PL/SQL Commands, page 8-91
- Defining Blackboard Commands, page 8-93
- Defining Forms Commands, page 8-93
- Defining Constant Commands, page 8-95
- Defining Delete Actions, page 8-95

Defining Java Commands

Use this procedure to define a Java command as an action in Script Author.

The command invokes a public Java method in a specified Java class, and returns the value returned by the invocation, if any, at runtime. The Java command object takes zero or more parameters and returns zero or one value.

Prerequisites

- Write the code that handles the data exchange.
- Make the Java class and method available to the Scripting Engine.

Steps

1. Navigate to a Command window.
2. In the Command Type area, select Java Command.
3. In the Command Info area, do the following:
    1. In the Name field, type the name of the command.
    2. In the Command field, type the command string as follows:

        ```
        <fully qualified class name>::<method name>
        ```

4. To list parameters for the command object, do the following in the Parameters area:
    1. Click Add.

        The Parameters window appears.

    2. In the Name field, type the name of the parameter.
    3. In the Value field, type the literal string value for the parameter.
    4. 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.

    5. Click OK to exit the Parameters window.
    6. To add another parameter, repeat steps 4.1 through 4.5.

5. Click OK to exit the Command window.

6. Save your work.

Guidelines

- The Command field specifies a fully qualified class path available to the Scripting Engine at runtime. Following this (and two colons) is the specific Java method invoked by the command at runtime.
- The class and its method are made available to the Scripting Engine by uploading an appropriately packaged class file from the Scripting Administration console.
- You can also reference Java methods already available to the Scripting Engine (for example, a class and method included in Oracle Scripting best practice Java methods).
- For information on packaging class files or uploading custom Java, refer to *Oracle Scripting Implementation Guide* and *Oracle Scripting Developer's Guide*.

References

- *Oracle Scripting Implementation Guide*
- *Oracle Scripting Developer's Guide*

Defining PL/SQL Commands

Use this procedure to define a PL/SQL command as an action in Script Author.

The command invokes a PL/SQL stored procedure or function in an Oracle database, and returns 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.
- Make the stored procedure or function available to the Scripting Engine.

Steps

1. Navigate to a Command window.
2. In the Command Type area, select PL/SQL Command.
3. In the Command Info area, do the following:

1. In the Name field, type the name of the command.

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

4. To list parameters for the command object, do the following in the Parameters area:

   1. Click Add.

      The Parameters window appears.

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

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

   4. 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.

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

   6. Click OK to exit the Parameters window.

   7. To add another parameter, repeat steps 4.1 through 4.6.

5. In the Connection area, specify the appropriate database connection for this PL/SQL command.

   To connect to the same database instance used when logging into Oracle Applications, select Use Login Connection.

   To connect to a different database instance, do the following:

   - Select Reuse an existing connection.

   - Select the appropriate existing connection from the connection list.

     Note: A database connection must be defined in the Connection List before you select it here. Use the Connection List window to create, modify, and test database connections for each script. For more information, see Working with Database Connections, page 8-86.

6. Click OK to exit the Command window.

7. Save your work.

Defining Blackboard Commands

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

To return a value, you must specify the blackboard key name. If returning a value from a question earlier in the script, ensure that the flow of the script necessitates the script end user to provide a response.

The key name for any Script Author panel question is the Name value entered into the Answer Entry dialog for a question in a graphical script.

Steps

1. Navigate to a Command window.
2. In the Command Type area, select Blackboard Command.
3. In the Command Info area, do the following:
    1. In the Name field, type the name of the command.
    2. In the Command field, type the name of the blackboard key name to be returned.

       If returning the answer to a panel question, the blackboard key name is the same as the question name. Question names are visible in a graphical script from the panel properties window. In the Panel tree, select Answers; each question defined for the panel is listed in the Answers pane.
4. Click OK to exit the Command window.
5. Save your work.

Defining Forms Commands

Use this procedure to define a Forms command as an action in Script Author.

This command invokes a PL/SQL procedure or function defined in an Oracle Forms PL/SQL package on the Oracle Forms server, and returns 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 window.
2. In the Command Type area, select Forms Command.
3. In the Command Info area, do the following:
    1. In the Name field, type the name of the command.
    2. In the Command field, type the command string.
4. To list the parameters for the command object, do the following in the Parameters area:
    1. Click Add.

        The Parameters window appears.
    2. In the Name field, type the name of the parameter.
    3. In the Value field, type the literal string value for the parameter.
    4. 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 parameters to another command must return a string value.
    5. In the In/Out list, select the parameters type.
    6. Click OK to exit the Parameters window.
    7. To add another parameter, repeat steps 4.1 through 4.6.
5. To define a return value, do the following in the Return Value area:
    1. Click Edit.

        The Parameters window appears.
    2. In the Name field, type the name of the return value.
    3. Click OK to exit the Parameters window.
6. Click OK to exit the Command window.
7. Save your work.

Defining Constant Commands

Constant commands return a constant value. This command object takes no parameters and returns one value.

A constant command can be used to display, at runtime, a value that the script developer determines in advance. For example, if using a script in the agent interface, you can populate a text field in the script information area with the name of a campaign by setting the campaign name as a constant. To do this, associate the constant command in a graphical script as a global script property, using the Command field in the Static Panel property of a script.

Alternatively, you can define a constant value as the default answer to a question. In such an example, define a constant command as the default command to the panel question in the question data dictionary.

Use this procedure to define a constant command as an action in Script Author.

Steps

1. Navigate to a Command window.
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:
    1. Click Edit.

       The Parameters window appears.
    2. In the Value field, type the return value.
    3. Click OK to exit the Parameters window.
5. Click OK to exit the Command window.
6. Save your work.

Defining Delete Actions

Note: The deletion that occurs during the execution of the Delete Action command does not display a popup window warning you that the record will be deleted. In general, if you have to delete records in scripts, Oracle recommends that you use PL/SQL business API's for processing deletes, in PL/SQL commands or in API blocks.

The delete action causes a table row to be deleted from the database, based on the information currently selected in the Scripting cursor.

When you execute a database query from a script using a query block, and one or more rows that match your criteria are returned, the cursor holds the first row. All rows are retained in static memory until the next query is executed (or until the interaction ends). Using the delete action causes the first row of the information stored in the cursor to be deleted from the database.

> **Note:** The delete action is not strictly a command, but is provided in the same interface to make this functionality accessible to script developers in a manner consistent with the application's existing paradigms.

Use this procedure to define a delete action. The delete action takes one value (the command name) and deletes the row selected in the cursor upon execution.

Steps

1. Navigate to a Command window.
2. In the Command Type area, select Delete Action.
3. In the Command Info area, type the name of the command in the Name field.

   For a delete action, the command name simply identifies the delete action within Script Author. Regardless of the name, the database row corresponding to the row held in the cursor is deleted upon execution of this action at runtime.
4. Click OK to exit the Command window.
5. Save your work.

Guidelines

- Scripting APIs are available to advance the cursor, check if the cursor is valid, and so forth. These can be used in combination with the delete action during the development of a script to determine which row of information should be deleted.
- The delete action will delete the row of information in the database last retrieved by a query, regardless of when in the current script interaction the last query took place. Thus, before including a delete action in a script, it is recommended to ensure that the relevant query is successful (that it will always return rows). Ensuring the validity of a query prior to executing the delete action will preclude deletion of a row returned for the previous query if the following query returns no values in a particular set of circumstances.
- For ease of maintenance, it is recommended that you insert the delete action as close as possible in the flow of a script to its relevant query.

- The delete action takes a single parameter in the Name field. The purpose of this field is to identify the delete action within the Script Author. Regardless of the name, the database row corresponding to the row held in the cursor will be deleted. The Name field can also be left blank, but this is not recommended.

References

Oracle Scripting Developer's Guide

Reusing Commands

The full range of Script Author commands (Java, PL/SQL, Blackboard, Forms, and Constant commands) can be defined in the command library, which stores Script Author commands in the applications database.

Once defined in the command library, commands can be copied and modified in the library, removed from the library, and imported into an existing or new script.

> **Note:** To make a command available for reuse in any script, you must first enter your command in the Script Author command library. If you define a command in a specific script instead of the command library, there is no method to copy or move that defined command to the command library.

You can perform the following tasks:

- Defining Reusable Commands, page 8-97
- Copying, Modifying, or Deleting Reusable Commands, page 8-98
- Importing Reusable Commands Into a Script, page 8-100

Defining Reusable Commands

Defining a command in the command library makes it available for reuse to script developers using Script Author in the same database instance. Reusable commands are defined using the Command Library window.

Use this procedure to define a reusable command.

Steps

1. From the Tools menu, select Command Library.

   The Command Library window appears.

   > **Note:** A database connection is required to access or store

commands in the database. If your session times out when performing actions in the command library, you will need to revalidate your current session.

2. To begin adding a command to the command library, click Add.

   The Command window appears.

3. Enter all command parameters.

   The value you type into the Name field in the Command info area is the name by which the command library will list your command.

4. Click OK in the Command window to save your work.

   The command is now stored in the library as a reusable command.

5. When you have completed your tasks in the command library, click Close to exit the Command Library window.

6. Save your work.

Copying, Modifying, or Deleting Reusable Commands

Once a command has been added to the command library, it can be copied, modified or deleted from the Command Library window.

Prerequisites

A reusable Script Author command must be stored in the applications database.

Steps

1. From the Tools menu, select Command Library.

   Note: A database connection is required to access or store commands in the database. If your session times out when performing actions in the command library, you will need to revalidate your current session.

   The Command Library window appears.

2. To *copy* an existing library command:

   1. Locate the appropriate command in the command list.

   2. Select the command.

3. Click Copy.

   Note: There may be a delay after this request as the command is retrieved from the database.

4. A copy of the selected command appears in the list. Appended to the Script Author command name are the additional characters "_Copy1".

3. To *edit* an existing library command:
   1. Locate the appropriate command in the command list.
   2. Select the command.
   3. Click Edit.

      Note: There may be a delay after this request as the command is retrieved from the database.

      The Command window for the selected command appears.

   4. Make the desired changes to the command and click OK to save your work.

      The updated command, as modified, is now stored in the library as a reusable command.

4. To delete an existing command from the command library:
   1. Locate the appropriate command in the command list.
   2. Select the command.
   3. Click Remove.

      A warning dialog appears, with the message "Remove this command from the library?"

   4. Click Yes to continue.

      The command library refreshes. The command you selected is no longer listed.

5. When you have completed your tasks in the command library, click Close to exit the Command Library window.

6. Save your work.

Importing Reusable Commands Into a Script

After a Script Author command is defined in the command library, it can be reused by any script developer accessing the same database instance. Use this procedure to import a defined command into the current script at a selected location.

Prerequisites

A reusable Script Author command must be stored in the applications database.

Steps

1. From the appropriate location in a graphical script into which you want to import an existing reusable command, navigate to a Command window.
2. Click Use Library Command.

   The Choose Command From Library dialog appears.
3. To import an existing library command:
   1. Locate the appropriate command in the command list.
   2. Select the command.
   3. Click OK.

      Note: There may be a delay after this request as the command is retrieved from the database.

      The Command window for the selected command appears. At this point, you can save the command in the script as it was imported, or modify the command as necessary.
4. Modify any command parameters or details if required.

   Note: Any modifications you make to the imported command are associated with the specific script into which you imported it, not the command library. To add new commands to the library, you must enter them directly from the Command Library selection from the Tools menu. To modify commands in the library, you must select the command from the command library and edit it directly.
5. Click OK to exit the Command window.
6. Save your work.

Deploying Scripts

You can perform the following tasks:

- Checking Script Syntax, page 8-101
- Deploying Scripts to the Database from Script Author, page 8-102
- Deploying Scripts to the Database from the Command Line, page 8-103

Checking Script Syntax

Graphical scripts must adhere to certain rules to be viable, executable scripts. These rules are validated, and the script compiled, each time you check the script syntax. Some example of these syntactical rules include the following:

- Each Script Author object evaluated during execution must have appropriate branching, from the start node to the termination node.
- Graphs in a script include, at minimum, the root graph. If you include any groups or blocks, these contain separate graphs (subgraphs). Each graph in a script must be properly terminated; in other words, each graph must include a termination node and appropriate branching.
- Each panel must have at least one question defined.
- Panels that use distinct branching must have one question selected as the default for distinct branching.

You can check a script's syntax at any time using the following procedure. Script syntax is also automatically checked each time you select the Deploy Script command.

Use this procedure to check the syntax of a graphical script. This procedure does not deploy the script.

Prerequisites

Create or open a graphical script.

Steps

1. Choose Tools > Syntax Check or click the Check Syntax icon in the toolbar.

   The Check Syntax icon is a graphic representation of a script with a checkmark.

   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 on a listed item to view specific contextual error messages (if

any).

On the lower portion of the canvas work area, an error pane appears, listing detailed information regarding each syntax error.

3. Fix any errors, and repeat this procedure until the message in the status bar indicates that the syntax check was successful, with no errors.

4. To clear syntax error messages and hide the error pane, right-click in the error pane until you see context-sensitive menu options, and select Hide or Clear.

   The error pane closes, providing a larger display area for objects on the canvas.

Deploying Scripts to the Database from Script Author

Deploying a script makes it available to the Scripting Engine for execution at runtime. After this step is successfully executed, the script can be executed using the Scripting Engine agent interface.

If a valid survey campaign deployment referencing this script is subsequently created and activated, you can also execute this script using the Scripting Engine Web interface.

Use this procedure to deploy an open graphical script from Script Author to the Oracle Scripting database schema.

Prerequisites

- You must have a syntactically correct script.
- You must be using the Script Author Java applet in a validated Oracle Applications session that has not timed out.
- Open a script to be deployed in Script Author.

Steps

1. Choose Tools > Deploy Script or click the Deploy Script to DataBase icon in the toolbar.

   The Deploy Script to DataBase icon looks like a database and a floppy disk.

2. If there are errors, then the Syntax tab is displayed and the Debug pane appears. (See Checking the Script Syntax, page 8-101.)

3. If there are no errors, the script deploys to the IES_DEPLOYED_SCRIPTS table in the Oracle applications schema.

   The status line reads "Deployment to database successful."

Guidelines

- Deploying a script to the applications database is required before you can execute a script in the Scripting Engine (using either interface).
- Immediately after deploying a script successfully, you can execute a script using the Scripting Engine agent interface.
- Before you can execute a script in a Web browser using the Scripting Engine Web interface, you must also administer survey campaign requirements and activate a deployment. For information, see Using the Scripting Engine > Using the Scripting Engine Web Interface, page 3-8.

Deploying Scripts to the Database from the Command Line

Deploying a script makes it available to the Scripting Engine for execution at runtime. After this step is successfully executed, the script can be executed using the Scripting Engine agent interface.

If a valid survey campaign deployment referencing this script is subsequently created and activated, you can also execute this script using the Scripting Engine Web interface.

Use this procedure to deploy a syntactically correct script file from the command line to the Oracle Scripting database schema.

Login and Responsibility

- You must have access to the applications server, and be able to execute a command line interface.
- There are no specific Oracle Applications responsibilities required for this task. However, you must have the apps username and password to execute the shell script and deploy the script from the command line.

Prerequisites

- This command invokes a shell script introduced in Oracle Applications release 11.5.10 or later or Interaction Center Family Pack R or later. You must be using an updated version of Oracle Applications to perform this task.
- AutoConfig must be executed to generate the shell script so that it is available for this task. This is a one-time step.
- You must have a syntactically correct script.
- You must have access to the shell script to execute it from the command line, and you must simultaneously be able to access the script file to deploy it.

Steps

1. Connect using Telnet to the enterprise system, or enter a shell by opening a DOS prompt.

2. Change to the APPL_TOP directory in the Oracle Applications file structure.

3. Ensure that your environment is set correctly.

   If necessary, change to the appropriate environment, ensuring that environment variables are set correctly.

4. Using FTP or another method, place the script file or files to be deployed in a location accessible to the shell script. For example, move the script file to the APPL_TOP.

5. For each script file to be deployed, execute the following command:

   Note: Use **cooper.sh** (as below) in your command for UNIX and Linux operating systems; use **cooper.cmd** if using a Windows operating system.

   ```
   $IES_TOP/admin/install/<ORACLE SID>/cooper.sh <fully
   qualified path of script file> <script file> <APPS
   username> <APPS password>
   ```

6. If the deployment is successful, the compiler returns a status code of 0, and you should see the following output:

   ```
   cooper.sh PASS: deploying <script file>
   cooper.sh exiting
   ```

7. If the deployment fails, the compiler returns a status code of 1, and you should see the following output:

   ```
   cooper.sh FAIL: deploying <script file>
   cooper.sh exiting
   ```

Guidelines

- Upon executing the shell script, a log file (cooper.log) is generated. If experiencing difficulties, access the log file in the path `$IES_TOP/log/cooper.log` for information.

- Possible reasons for failure include incorrect script file name, invalid script (not syntactically correct), incorrect apps-level username, or incorrect apps-level password.

Recovering from an Expired Session

Script Author is a Java applet. This applet is accessed only from a validated Oracle Applications session. When using Script Author, if you leave the application idle, the session can expire when the ICX Session Timeout threshold is exceeded. This causes a timeout for all of Oracle Applications. The user must re-authenticate the Oracle Applications session, as described below.

The amount of time before a session expires is based on environmentally dependent variables (specifically, the ICX Session Timeout system profile option).

You can recover from an expired applications session without losing work, by following the procedure below. If it is your intention to publish the script directly to the database, Oracle recommends that you save the script to a local or network file system temporarily, until you reestablish the script session.

Use this procedure to recover from an ICX Session timeout experienced from a Script Author session.

Prerequisites

You must be in a Script Author session that has timed out, as evidenced by an error message indicating "Your Script Author session has expired."

Steps

1. When you receive the error message indicating that your Script Author session has expired, click OK.

   The Error Message window closes. The Script Author canvas is visible. The message area below the canvas contains a message about the session timeout.

2. If you have changes, save the script to the network or local file system, page 8-15.

3. Leaving the Script Author Java applet window open, return to the Scripting Administration console (which is open in a Web browser window).

4. From the Scripting Administration console, click Home.

   The session expired login page for your environment appears.

5. Re-authenticate your Oracle Applications session in the same manner you did to start the now-expired session.

   For example, if you entered a username and password, reenter that information now.

   Note: You must use the same user account (user name or user ID)

in order to re-authenticate an existing session.

The login window is replaced by the default responsibility for your current Oracle Applications user account.

- If your default responsibility is Scripting Administrator, the Scripting Administration console appears.
- If you have a different default responsibility, navigate through Oracle Applications and select the Scripting Administrator responsibility.

  The Home tab is visible, and your Oracle Applications session is restarted.

6. Click Launch Script Author.

   A new Web browser window opens.

   The Script Author applet window appears and has focus.

7. Continue your work.

Packaging Java Bean or Custom Java Code Into a JAR File

Oracle Scripting*i* allows the substitution of a custom Java Bean user interface in place of a standard Scripting panel.

Oracle Scripting*i* also supports the use of custom Java methods in a script, when the script appropriately references the method using a Script Author command.

In order for a script referencing custom Java (a Java bean or a Java method) to supply that code to the Scripting Engine user interface at runtime, the Java code must first be packaged into a Java Archive (JAR) or WinZip (ZIP) file. Oracle recommends using JAR file format.

Due to a limitation in JDK 1.1.8, there are only two supported methods of packaging a JAR file appropriately. Use either of the following methods to appropriately package custom Java code for execution at runtime.

> **Note:** While an entire panel can be substituted with a Java bean, substituting only a panel *answer* (or panel node) with a Java bean is no longer supported in Oracle Scripting release 11.5.7 and later, due to the WYSIWYG editing Script Author feature.

The two methods are:

1. Use the command-line "jar" utility from JDK 1.1.8 and specify no compression. For example:

`jar -cf0 TestBean.jar ...`).

This method is described below.

2. Create the custom JAR file in standard .ZIP or .JAR format (with any PC-standard compression utility) with or without compression. If that utility saves the resulting archive in .ZIP format, then simply rename the file extension from .ZIP to .JAR.

   For enterprises making extensive use of custom Java with Oracle Scripting, this method (specifying compression) is recommended to conserve database table space. This method of packaging a JAR file may differ based on the compression utility used. For specifics, see the compression utility manufacturer's documentation.

Prerequisites

- As with all custom code, the Java bean or Java method must be created by a certified Java developer using any appropriate Java development tool.
- Oracle recommends compiling custom Java code in support of Oracle Scripting using Java Development Kit (JDK) 1.1.8.

Steps

1. Write the Java source code for the bean or Java method.
2. Compile into one or more .class files using any appropriate method.
   - When using custom Java for Oracle Scripting in the agent interface,, this code must be compiled using JDK 1.1.8.
3. Copy or move compiled Java class files into the directory in which you have JAR.EXE (for example, C:\jdk1.1.8\bin; your directory may differ).
4. Open a Command prompt (Start > Run > CMD)
5. Change to the appropriate directory. For example:

   ```
   cd jdk1.1.8\bin
   ```

   At the command line, start the JAR utility. For example:

   ```
   C:\jdk1.1.8\bin>JAR.EXE).
   ```

6. Execute the command to include one or more specified class files and include them into a Java Archive, specifying no compression.

Guidelines

The general syntax for the jar command is as follows:

```
jar <option string> <jar file> <manifest file> <input files>
```

The option string specifies actions to be performed by the jar utility and is always required. Options are detailed below.

- The jar file parameter identifies the name of the file to be created by executing the command.
- A manifest file identifies a file describing the contents of an archive. Specifying a manifest file is irrelevant for the purposes of creating JAR files in support of Oracle Scripting.
- Input files specify one or more Java beans or class files to be included in the JAR file resulting from executing the command. If multiple input files are specified, separate each in the jar command by one space.

JAR Options

Options for the jar utility include the following:

Code	Description
-c	Creates a new archive using the input files.
-t	Lists a table of contents for the archive.
-x	Extracts files from an existing JAR file.
-u	Extracts files from an existing JAR file.
-v	Generate verbose output on standard error
-f	Specify archive file name
-m	Include manifest information from specified manifest file
-0	Store only; use no ZIP compression
-M	Do not create a manifest file for the entries

Examples

To add a single class file (onefile.class) into a Java archive called destination.jar, type:

```
C:\jdk1.1.8\bin> jar -cf0 destination.jar onefile.class
```

To combine two class files (file1.class and file2.class) into a Java archive called destination.jar, type:

```
C:\jdk1.1.8\bin> jar -cf0 destination.jar file1.class file2.class
```

After creating your JAR files and deploying them as appropriate, remove any unnecessary files you made or copied to avoid confusion.

9

Using the Scripting Engine

This chapter covers the following topics:

- Introduction
- Using the Scripting Engine Agent Interface
- Using the Scripting Engine Web Interface
- Scripting Engine Sessions and Transactions

Introduction

The Scripting Engine is the component of Oracle Scripting that executes Script Author scripts at runtime. It is essentially a collection of base Java classes that process a script, any custom code associated with the script, and script end user interactions.

Scripts can be executed in one of two Scripting Engine interfaces: the agent interface (a Java application running in an Oracle form), and the Web interface (in which a script executes in an Oracle Applications-compatible Web browser). In either interface, the Scripting Engine displays text, images, questions and prompts (in units known as panels). Script end users provide answers to the one or more questions in each panel by clicking buttons, selecting choices from lists, or completing text fields. To progress from one panel to the next, the user clicks a submit button (usually labeled Continue in the agent interface or Next in the Web interface).

Each Scripting Engine interface interprets end user responses and custom code. The progression through the script (script flow) may be determined dynamically, if branching logic is included in the script.

Panels are the only Script Author objects that display in a script. Other objects control processing, but do not display.

Panels are displayed in the agent interface in a panel display area. Other components unique to this interface include a script information area (optional), a shortcut button area (optional), and a progress area (which shows the path through the current script session and, optionally, lists the responses selected for each question per panel). These

components are all wrapped in a script frame that contains a progress indicator, a Disconnect button, and (optionally) a Suspend button.

In the Web interface, each panel is represented by one JSP page. This is equivalent to the panel presentation area in the agent interface. The frame around the panel contains Back, Next, and Reset to Default buttons; for suspendable scripts, it also includes a Save for Later button. No other components display, other than the Web browser's own interface controls.

This section includes the following topics:

Using the Scripting Engine Agent Interface

Scripts created in Script Author are deployed to the applications database. Thereafter, interaction center agents with the appropriate Oracle Applications responsibility can launch a script and run it in the Scripting Engine agent interface.

Oracle Scripting is integrated with the following business applications:

- The Customer Support module of Oracle TeleService
- Oracle TeleSales

> **Note:** In previous releases, Oracle Collections was integrated with Oracle Scripting.

In production, scripts are typically executed from within one of these applications. For script testing purposes, scripts can also be launched in standalone mode (using the Script User or Scripting Agent responsibility). Standalone mode is not supported for typical interaction center operations.

This section includes the following topics:

References

- For information on launching scripts from each business application, refer to the

Integration section of *Oracle Scripting Implementation Guide*.

- For descriptions of each element of the Scripting Engine UI, see Understanding the Scripting Engine, page 3-1.

Launching a Script in Standalone Mode

Use the following procedure to launch a script in the Scripting Engine agent interface in standalone mode.

> **Caution: Standalone mode (executing a script without the use of a business application) is intended for script testing only, and is otherwise not supported by Oracle.**

Prerequisites

- A script must be deployed to the applications database using Script Author.

Login

Log into Oracle applications using the Personal Home Page login, or the Single Sign-On login if implemented.

Responsibility

Scripting Agent, Vision Enterprises

Scripting User

Steps

1. From the Navigator, select Scripting Demo Form and click Open.

   The Script Chooser window appears.

   The Oracle Forms Developer window entitled Oracle Scripting is referred to as the Script Chooser. This window provides access to a list of all active scripts created and deployed to the applications database using Script Author.

2. In the Script Chooser window, from the Script Name/Language list, select the combination of script name and language for the script you want to launch.

3. Click Start Scripting.

   The Script Chooser minimizes, and the script launches in a separate window.

Executing a Script in the Agent Interface

Use the following procedure to execute a script in the Scripting Engine agent interface.

Prerequisites

- A script must be deployed to the applications database using Script Author.
- Launch a script.

Login

Log into Oracle applications using the Personal Home Page login, or the Single Sign-On login if implemented.

Responsibility

Customer Support

Telesales Agent

Scripting Agent, Vision Enterprises *(for testing only)*

Scripting User *(for testing only)*

Steps

1. From the first panel, review all information displayed.
2. If there is only a single question control (a submit button) in the panel, then click the button (or select the space bar when the button control is selected).

   For example, if the panel contains a Continue button, click Continue.

   The script progresses to the next panel, if any. If the script transaction is complete, the script frame closes.
3. If the question user interface (UI) control is a series of submit buttons, then select and click the appropriate button. To select the button using the keyboard, press the space bar. To cycle among two or more choices, use the Tab key.

   The script progresses to the next panel, if any. If the script transaction is complete, the script frame closes.
4. If the panel contains two or more question controls, then review and select all question responses as appropriate.
   - If the question user interface (UI) type is a text field, text area, or password field, then type in the field, as appropriate. An absence of a response (leaving a null value) is acceptable; unless validation is explicitly associated with the panel question, you are not required to enter a value into any of these question UI types.
   - If the question UI type is a radio button or a drop-down list, you must select one of the answer choices displayed by the question control.

- If the question UI type is a check box or a checkbox group, select each option (answer choice) as appropriate. For a single check box, select the check box value if appropriate. For a checkbox group, select as many or as few answer choices as appropriate. Leaving a check box clear (passing a null value) is acceptable.

- If the question UI type is a multi-select list box, you may select no values (null), one value, or any combination of the values displayed. Clicking on any answer choice option in the list selects that single item. Holding down the Control key (on a Microsoft Windows platform) or the Option key (on a Macintosh OS platform) allows you to select any two or more answer choices. Clicking the Shift key (on any platform) allows you to select any two or more consecutive answers.

- If the question user interface (UI) type is a series of submit buttons, then select and click the appropriate button. To select the button using the keyboard, press the space key. To cycle among two or more choices, use the Tab key.

  When you have provided the appropriate answers to all questions in a panel, then click Continue.

5. Optionally, to hide responses provided thus far during the current script transaction, then in the Progress area, click Hide Answers.

6. Optionally, if responses provided thus far during the current script transaction are hidden, then if you want to display those answers in the Progress area, click Show Answers.

7. Repeat steps 2 through 4 for each panel in the script.

8. To navigate to the previous panel in the script, from the application toolbar, click Previous Panel.

   Oracle Scripting displays the appropriate panel.

9. If you have navigated backward in a script, then to return to the next panel, from the application toolbar, click Next Panel.

   Oracle Scripting displays the appropriate panel.

10. To navigate to any panel you have already visited in the script, then from the progress area, scroll if necessary to the appropriate panel name, and click in the progress area.

    Oracle Scripting displays the appropriate panel.

11. If you have navigated backward in a script, then to return to the last panel you have viewed thus far in the script, from the application toolbar, click Last Panel.

Oracle Scripting displays the last panel displayed in the current script transaction.

12. Optionally, to execute a function contained in a shortcut button, click the button.

    The function executes.

13. Optionally, to exit the script, click Disconnect.

    The script jumps to the first panel in a group designated with the WrapUpShortcut. If this group contains no panels, then the script transaction ends, and the script frame closes.

14. Optionally, if enabled, then to suspend the script transaction, click Suspend.

    The script transaction is saved as a suspended script, and the script frame closes.

Suspending an Oracle Scripting Transaction in the Agent Interface

Use the following procedure to suspend a current Oracle Scripting transaction running in the Scripting Engine agent interface.

> **Note:** The Oracle application that launches the script in the agent interface controls whether or not the script can be resumed. Check the documentation for the appropriate Oracle application to determine if the resume feature has been implemented.

Prerequisites

- The Suspendable global script property must be selected in the deployed Script Author script.
- The IES : Display Suspend Button on Script Frame profile option must be set to True.

Login

Log into Oracle applications using the Personal Home Page login, or the Single Sign-On login if implemented.

Responsibility

Customer Support

Telesales Agent

Steps

At any point of time during the runtime session, you can click the Suspend button.

Any answers you have entered in the panel where you clicked the Suspend button are

saved, as "intermediate" answers, but not submitted to the Scripting Engine. The intermediate answers are available to you when you resume the suspended transaction.

Using the Scripting Engine Web Interface

Scripts created in Script Author are deployed to the applications database. Thereafter, a script can be designated as the survey questionnaire for a survey campaign in the Survey Administration console. When the survey campaign has a cycle and deployment, and the deployment is activated, then a valid survey URL results.

The Scripting Engine Web interface provides the ability to execute Script Author scripts in an Oracle Applications-compatible Web browser (over the Internet or on an intranet, including across secure HTTP). This is sometimes known as the survey runtime component of Oracle Scripting.

Scripts are executed in the Scripting Engine Web interface using one of two methods, based on the deployment type. *Standard deployments* require only survey campaign administration. Networked users of a Web browser can then launch and execute a script in a Web browser by accessing the URL.

Targeted deployments require additional administration of a marketing list (using Oracle Marketing Online functionality) and of an invitation or reminder master document (using Oracle One-to-One Fulfillment functionality). This assigns each list member a unique respondent ID, which is included in the valid survey URL for targeted deployments.

When a list member receives an e-mail inviting him or her to use the script, they simply access the URL, and execute the script in a Web browser.

Users of Oracle self-service Web applications such as Oracle iSupport can execute a script in a Web browser by clicking the survey URL from within the application. These require only standard deployments.

A script executed using the Scripting Engine Web interface is referred to as a survey questionnaire (or a survey) if its purpose is simply to solicit information or feedback. Otherwise, it is referred to as a Web script. Both terms are used interchangeably to refer to execution of a script in a Web browser.

This section includes the following topics:

Launching Scripts in the Web Interface

This section includes the following topics:

Launching a Script from an Oracle Self-Service Web Application

Prerequisites

- You must have a valid survey URL for an activated deployment.
- Your web application must be customized to embed the survey URL in the user interface.

Login

Log into Oracle Self-Service Web Applications. Use the Single Sign-On login if implemented.

Responsibility

The appropriate responsibility is determined by the integrated Web application. For example, to launch scripts from Oracle iSupport, log into Oracle iSupport using the iSupport User responsibility.

Steps

1. From the appropriate application, access the page on which the link to the survey URL is located.
2. Click the survey URL link.

   The Web script or survey executes, hosted in the user interface of the logged-in application. Upon completion of the script, your Oracle Applications session is still valid, and you can perform work in the application as required.

Launching a Script from an Invitation or Reminder

Prerequisites

- You must be a member of an Oracle Marketing list.
- You must be the recipient of an e-mail message (invitation or reminder) originating from an Oracle One-to-One Fulfillment master document, containing a valid survey URL, including a unique respondent ID for your list record.

Login

None

Responsibility

None

Steps

1. On a networked computer connected to the Internet, open the invitation or reminder e-mail message.

2. From the invitation or reminder e-mail message, application, click on the survey URL.

   Alternatively, copy the survey URL, and paste it into the Location or Address field of your Web browser, and press the Return key.

   The Web script or survey executes. You are logged in as an applications guest user. The only action you can perform is to complete the survey. Upon completion of the script, your Oracle Applications session is terminated.

Launching a Script from a Known URL as a Guest User

Prerequisites

- You must know the URL of an active survey deployment.

- To launch the script as a guest user, you must not be in an active session of an Oracle Applications session, or have a valid cookie for a Web browser with valid Oracle Applications session authentication information cached. For example, if you are in a valid session using Microsoft Internet Explorer, open Netscape Navigator to execute the Web script or survey.

Login

None

Responsibility

None

Steps

1. On a networked computer connected to the Internet, enter the URL of a valid survey URL into the Location or Address field of your Web browser.

2. Press the Return key.

   The Web script or survey executes. You are logged in as an applications guest user. The only action you can perform is to complete the survey. Upon completion of the script, your Oracle Applications session is terminated.

Navigating the Web Interface

Important: You are strongly advised to use only the Oracle Scripting Back and Next buttons to navigate between the panels of a script in

the Web interface.

If you use the native browser "Back" and "Next" buttons to progress through Web interface script sessions, while there are some situations where the Scripting Engine can interpret and act upon your panel answers, there are others where it cannot. For more information, see Issues Arising from the Use of Browser Buttons, page 9-11.

This section consists of the following topics:

General Navigation

Your general progress through a script is as follows:

- You enter answers on a panel, and when you are satisfied that they are correct, you click Next. The next panel in the script sequence, as determined by the Scripting Engine, appears.
- If you need to change answers in a panel that you previously visited in the current session, click Back. Continue to click Back until you reach the panel containing the answers that you want to change.

You can think of using the Next button as going forward through the script, and using the Back button as a way to temporarily pause the standard forward script progression.

Note: Scripts running in the Web interface will generally *not* contain the Continue button, that is designed to appear in panels when you run scripts in the agent interface. If the Continue button appears in a panel during a Web interface session, you can click either the Oracle Scripting Next button or the Continue button - they will behave identically.

In this section dealing with navigation in the Web interface, any reference to the Next button applies also to the Continue button.

Using Button Panels

There are special considerations for button panels, that is, panels with one question and one or more push buttons that serve as the answers.

The first time that a button panel is displayed, the Next button will be disabled, and you must click one of the push buttons to answer the panel.

If you return the script to the same push button panel, then the Next button is enabled. To progress forward through the script, you can then perform either of the following operations:

- Click any of the push buttons in the panel - possibly changing the answer
- Click the Next button - equivalent to clicking the same push button that was answered the last time

Submitted and Saved (Intermediate) Answers

To allow for maximum flexibility when you change previously-entered answers or resume a script that you previously suspended, Oracle Scripting has the concept of *submitted* and *saved* answers.

When you click the Next button (or a button in a button panel), the current panel's answers are submitted to the Scripting Engine, which processes them - together with any related branches, blocks, groups, and actions designed in the script - and determines the next panel to be displayed.

Clicking the Back button displays the previous panel in the session, clicking the Save for Later button suspends the current session. In both cases, the current panel's answers are saved, but not submitted to the Scripting Engine.

Answers are saved or submitted only if you use the Oracle Scripting Back, Next, and Save for Later buttons, and never when you use a browser button.

Saved answers are only submitted to the Scripting Engine when you click the (Oracle Scripting) Next button, or a button in a button panel.

> **Note:** Oracle Scripting stores both submitted and saved answers in xml form in the database. Because the saved answers have not yet been processed, they can be thought of as "intermediate" answers. When referring to answers, the terms "saved" and "intermediate" are used interchangeably in this section.

Issues Arising from the Use of Browser Buttons

Oracle recommends that you do *not* use native browser Back and Next buttons in Web interface script sessions: you should always use only the Oracle Scripting Back and Next buttons.

If you use both the Oracle Scripting and the browser buttons, this can lead to the situations described in the following sections.

Getting to Invalid Pages

You have used the Oracle Scripting Next *(OS-N)* and Back *(OS-B)* buttons to visit and revisit a panel, such as panel B in the sequence A -*(OS-N)*-> B -*(OS-N)*-> C -*(OS-B)*-> B -

(OS-N)-> D.

The first time you click *(OS-N)* on B, Panel C appears; the second time that you visit B, you change an answer, click *(OS-N)*, and now D appears.

The net effect is that C is no longer in the script flow.

If, from D, you click the *browser* Back button twice to get back to C, then click the Oracle Scripting Next *(OS-N)* button, you will get a message indicating that you have reached a page that is no longer valid: after acknowledging the message, you will be returned to the last valid page, that is, D.

Losing Saved Answers

When you click the Oracle Scripting Back *(OS-B)* button while on a panel, all the answer changes that you have made on that panel are saved, but not submitted to the Scripting Engine.

For example, suppose you enter data in and progress through the panels A, B, C, D by clicking the Oracle Scripting Next *(OS-N)* button each time.

Now, you click the Oracle Scripting Back *(OS-B)* button twice to return to panel B. On panel B, you make some changes, then click the Oracle Scripting Back *(OS-B)* button to get back to panel A. The changes you made on panel B are saved as "intermediate" answers, but not submitted to the Scripting Engine.

If you now use the *browser* Back button to return to panel C, make some changes, then click the Oracle Scripting Next *(OS-N)* button, Oracle Scripting will process the answers entered on panel C, but warn you that there were previous panels with unsubmitted answers that were not used.

Suspending and Resuming Web Interface Oracle Scripting Transactions

The facility to suspend and resume Oracle Scripting transactions in the Web interface applies mainly to targeted surveys. Oracle applications that launch standard surveys may implement the facility to allow users to suspend and resume scripts launched from those applications.

Prerequisites

To be able to suspend and resume a script in the Web interface, the script must belong to a suspendable survey campaign. Only if that is true will a Save for Later button appear on the screen, when the script is run in the Web interface. There is no further dependency on any profile option.

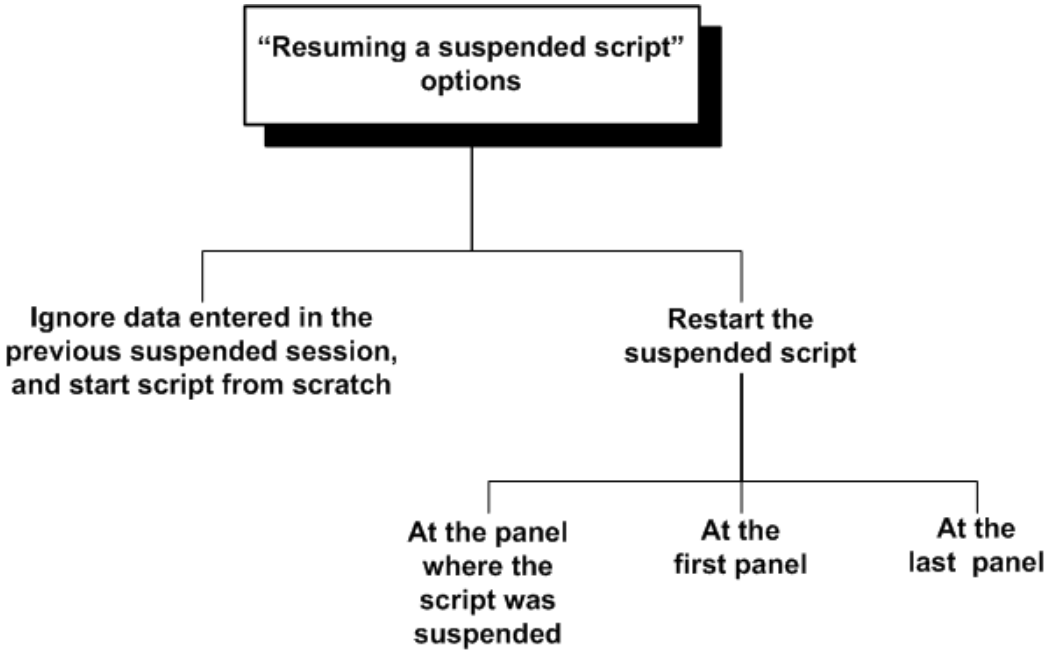
Suspending the Script

To suspend the script, click Save for Later. All the script answers entered up to that point are stored, including the answers entered by the user on the final displayed page.

Resuming the Script

Users of suspended targeted surveys can resume a suspended survey by clicking the same link that they received by e-mail originally. Oracle Scripting will then display a page showing the date and time that the survey was suspended, and a message asking if the user wants to start a new survey transaction from scratch or to resume the saved survey by restarting the suspended script.

The general options available to the user on script resumption are as appear in the diagram following.



If the user chooses to start the script from scratch, all of the data entered in the previous suspended session is ignored; the user will be entering completely new data.

Users who do not want to start the script from scratch will be able to restart the script at a particular panel, with Oracle Scripting reconstructing the data using the answers entered during the previous session or sessions. The option that determines the "restart" panel is controlled and set by the application that creates the e-mail URL. The restart panel can be either the panel where the script was suspended, the first panel of the script, or the last panel accessed in the script flow.

> **Note:** Assume the panel flow for a script session was A, B, C. In panel C, you clicked the Oracle Scripting Back button to re-enter data on panel B, then clicked Save for Later. On script resumption, the "last page" option restarts you at panel C; the "panel at which the script was suspended" option restarts you at panel B.

> **Note:** For more information about when and how Oracle Scripting stores answers to panel questions, see Submitted and Saved (Intermediate) Answers, page 9-11.

Error Handling During Resumption

During the resume process, in reconstructing the script, the Scripting Engine processes the answers that were given in the original transaction, and verifies that each panel returned is the same as in the original transaction. Two types of error may occur during the resumption, recoverable and unrecoverable.

Recoverable errors are generated when calling custom code, such as Java or PL/SQL, and do not disrupt the script flow. You will be given the opportunity to ignore recoverable errors and continue processing the script resumption.

> **Note:** Recoverable errors that occurred and were ignored in the original transaction can trigger new messages during the resume process; that is, errors seen during the script resumption do not have to be specific to the current resume process.

Unrecoverable errors are those that prevent the Scripting Engine from processing further. Situations where this may occur are the following:

- Due to external changes, none of a panel's branches are valid.
- A different panel is returned during the script resumption from the panel in the original transaction.

To deal with errors during script resumption, you can select from the following options *(the first two are only for recoverable errors)*:

- Ignore the recoverable error and attempt to continue the resume process.
- Ignore *all* recoverable errors and attempt to continue the resume process.
- Cancel the resume transaction.

  The information for the script is restored back to the saved state, as if no resumption had been attempted.
- Restore the transaction back to the point at which the error occurred.

  Users can then continue the script from that point.

Scripting Engine Sessions and Transactions

This section includes the following topics:

Oracle Scripting and Oracle Applications Sessions

Each execution of a script must occur within the context of a valid Oracle Applications session.

For users of the agent interface, the agent user must log into Oracle Applications before starting an Oracle Scripting session.

For uses of the Web interface, scripts can be executed from an Oracle self-service Web application. If so, the current Oracle Applications session validation information is used. This session is governed by ICX session timeout settings.

If executing a script in the Web interface without an existing, valid Oracle Applications session, the applications guest user account is used to begin an Oracle Applications session. This session terminates upon completion of the script transaction. No other work in any other Oracle application can occur.

For more information, see *Oracle Scripting Implementation Guide*.

Oracle Scripting Sessions

All script transactions are executed using the Scripting Engine component of Oracle Scripting, and each script requires an Oracle Scripting session. This session is dependent on an existing Oracle Applications session.

The behavior of Oracle Scripting sessions differs based on which interface of the Scripting Engine is used to run the script.

For scripts executed in the Scripting Engine agent interface, each Oracle Scripting session can contain one or more transactions.

For scripts executed in the Scripting Engine Web interface, each Oracle Scripting transaction uses its own session.

Oracle Scripting Sessions in the Scripting Engine Agent Interface

The first time in an Oracle Applications session that the Scripting Engine agent interface is invoked (by a launch script request), an Oracle Scripting session starts.

The session is dependent on Oracle Forms. In an Oracle Applications session, Oracle Forms runs as an applet on the client desktop. When you first log in and request a script to launch, the Oracle Scripting session begins. An Oracle Scripting session remains open until it expires, or until the Oracle Applications session is terminated (in other words, the user logs out of Forms-based applications).

The Oracle Scripting session times out if you are idle (from a Scripting Engine perspective) for longer than a specified time. Thus (whether you are idle in the middle of a script transaction, or if you are between script transactions), if you do not perform any work in the script (or request a new script to execute for the duration of time specified in the session idle timeout property), this session expires. The Apache session idle timeout is a servlet property, established in the ZONE.PROPERTIES file. (For more information, see *Oracle Scripting Implementation Guide*.)

After a session is started, you can execute any number of scripts in the Scripting Engine agent interface. Normal rules apply; for example, you cannot launch two scripts simultaneously, nor start a script from within another script.

Oracle Scripting Sessions in the Scripting Engine Web Interface

Each time a user requests a script to execute in a Web browser, a new Oracle Applications session starts. The script transaction executes within the Oracle Scripting session. When the transaction ends, the session also ends. This is true:

- Regardless of how often or how frequently a script is requested to execute in a Web browser.
- Regardless of whether the script is executed as a survey or Web script.
- Regardless of whether the script uses an existing authenticated Oracle Applications user, or the applications guest user.
- Regardless of whether the script is hosted in the UI of an Oracle self-service Web application or run in "standalone" mode in its own browser window.

Like all scripts executed in this interface, an existing, deployed, validated script is prerequisite, as is the existence of a valid, active survey campaign. Each transaction (and each session) is accessed by the end user accessing the appropriate survey URL (whether through clicking a hypertext link, or an image associated with a hypertext link).

When using the Scripting Engine Web interface in a hosted mode (when executing a script from the UI of a self-service Web application), you can execute as many script transactions as required for the business purposes of the self-service application (with each script transaction occurring within its own Oracle Scripting session). In contrast, when in an interaction or transaction of a Forms-based business application, you can only launch a single script per business application transaction.

Like scripts executed in the agent interface, the Scripting session times out if you are idle within the script longer than the amount of time specified in the session idle timeout parameter on the Apache server.

The options available to you after the timeout depend on whether or not the survey campaign script is in a suspendable survey campaign.

If the script is in a suspendable survey campaign, the following considerations apply:

- When the session times out, the script is suspended.
- Users of suspended targeted surveys can resume a suspended survey by clicking the same link that they received by e-mail originally.
- In general, you can resume a suspended script in the Web interface if the necessary features are built into the application that launched the script.

If the script is not in a suspendable survey campaign, the script cannot be resumed at the panel where the timeout occurred: you will have to relaunch the script in a new browser window.

Script Transactions

Each time a script is launched by the Scripting Engine, an Oracle Scripting transaction begins. Each transaction ends upon standard completion of the script.

From a Wizard script perspective, standard completion of a script transaction occurs (from the script perspective) after the end user exits any panel in the flow of a script for which the next destination specified in the Script Wizard is "End script." When the end user provides one or more responses (as appropriate) to that panel, and the final page is then displayed in the Web browser, then any post-actions associated with the global script are then executed, and the script transaction then ends.

From a graphical script perspective, standard completion of a script refers to reaching the termination node on the root graph. Any post-actions associated with the completion of the global script are then executed, and the script transaction then ends.

In an appropriately configured graphical script, any Disconnect or WrapUp group (a group on the root graph that contains the WrapUpShortcut shortcut name) is connected to a termination node on the root graph. Thus, in the agent interface, clicking the Disconnect button results in directing the session to the group with the WrapUpShortcut property. Upon completion of the flow in this group (if any), the script transaction ends.

There is no equivalent for the Disconnect button in the Web interface.

Clicking the Suspend button in the agent interface, for the purposes of this discussion, is considered standard (if temporary) completion of a script session. Clicking the Save for Later button in the Web interface suspends the transaction.

Upon standard completion of a script, when the Continue or Next button is clicked by the end user in the last panel of the script, the Scripting Engine performs all post-panel processing, and then performs all standard script wrapup functions.

> **Note:** When you reach a termination point in a script executing in a survey campaign designated as Always Suspend, the script is suspended, and the Scripting Engine performs post-panel processing and script wrap-up functions.

Post-Panel Processing

The following occur for each panel when the Continue button is clicked by the end user of any script:

- The responses provided for the panel are sent to the Scripting servlet.
- If any questions in the panel have validation associated with them, those validation commands execute.
- Outgoing branches that determine the next destination in the flow of the script is evaluated.
- Post-action commands associated with the panel are executed.
- Action commands associated with the appropriate outgoing branch are executed.
- If the destination of the appropriate outgoing branch is an object that is processed but not displayed (a group or block), processing for that object then occurs, beginning with any pre-action commands associated for that object.

  If the destination of the appropriate outgoing branch is a panel, then any pre-action commands associated for that panel are executed, and then the panel is displayed.

Script Wrap-Up Functions

When the final destination is the end of the script (in other words, when the appropriate outgoing branch leads to a termination node), the following also occur:

- In the Oracle Scripting schema, the script end time is recorded in the database.
- If answer collection is enabled for the script, the answers provided by the script end user are recorded to the appropriate tables in the database.
- If footprinting is enabled for the script, the panels accessed during the script transaction, and the duration of time spent in each panel, is recorded to the database.
- If information is specified to be saved to any custom tables, this information is then recorded to the database.
- The transaction, and any PL/SQL operations or Delete actions occurring during the script transaction, is committed to the database.
- If using the Scripting Engine agent interface, the script window in which the script appears is minimized or hidden from view.

Premature Script Termination

A script session can be terminated prematurely by the script end user. Scripts can only

be prematurely terminated when a panel is displayed in the Scripting Engine user interface and the application is awaiting an end user response.

- In the agent interface, if the user directs the scripting window to close, and confirms the warning message, the script transaction is terminated as incomplete. The script session is retained.

- In the Web interface, if the Web browser window is closed before the last panel in the script is displayed, the consequences depend on whether or not the script is in a suspendable survey campaign. If it is, the script transaction is suspended; if it is not, the script transaction is terminated upon reaching the session idle timeout specified on the Apache server, and the script session is also terminated at that time.

If a script transaction is prematurely terminated:

- Panel post-actions (if any) are not processed.

- Post-actions of any groups or blocks containing the panel from which the script was closed are not processed.

- Post-actions of the global script (if any) are not processed.

Note, however, that after a session is prematurely terminated, the following still occur:

- Footprinting and answer collection (if specified) are still recorded to the database.

- Any operations performed on the database connection so far are committed.

A

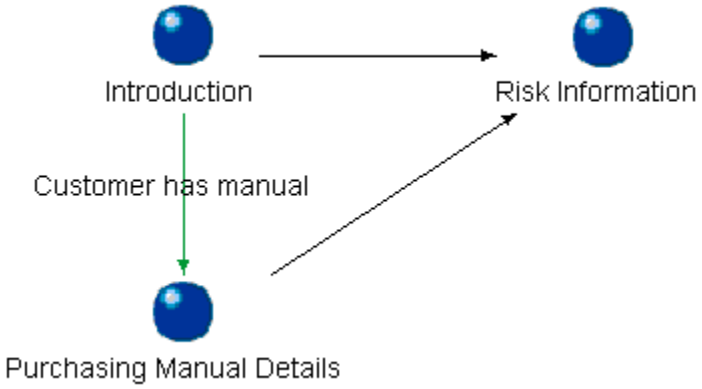
Example of Starting to Create a Wizard Script

Introduction

This appendix consists of the following topics:

Background

Initially, the Purchase and Risk script asks for the customer name, and whether the customer has a purchasing manual. Customers who have purchasing manuals are then asked for details of their purchasing manual. All customers are asked to provide risk information.



Purchase and Risk Script Panels

The panels for the script are the following:

- Introduction Panel, page A-2
- Purchasing Manual Details Panel, page A-2
- Risk Information Panel, page A-3

Introduction Panel

Please answer the following questions.

Customer Name _____

Do you have a purchasing manual?

- Yes
- No

Purchasing Manual Details Panel

Please answer these questions about your purchasing manual.

What content is described in your purchasing manual? (check all that apply)

- Procedures on purchase requisitions
- Procedures for purchase orders
- Procedures for approval for payment

How often is the purchasing control content updated?

- Each time a procedure changes
- Quarterly
- Annually

Risk Information Panel

Please answer the following question.

At which level is ownership of risks established?

- Department
- Site
- Individual person

First Steps of Setting Up Purchase and Risk Script

This section is designed to illustrate the main pages involved when you start to create the Purchase and Risk Script using the wizard.

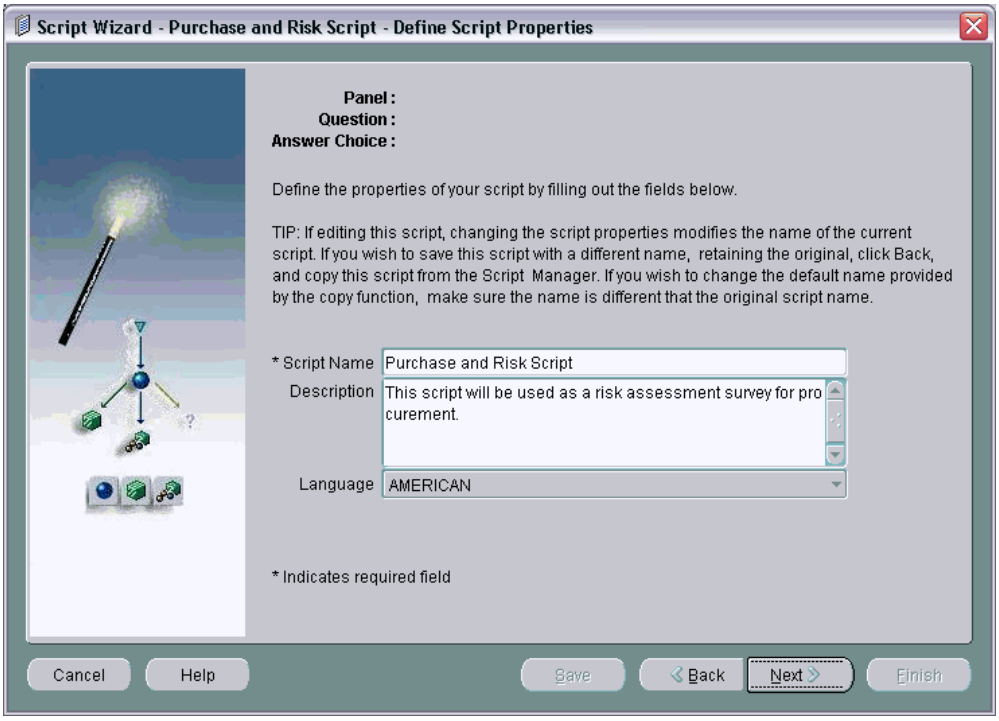
The topics in this section are as follows:

Defining the Script

1. Enter the following in the **Define Script Properties** page:

Column	Value
Script Name	Purchase and Risk Script
Description	This script will be used as a risk assessment survey for procurement.

Column	Value
Language	AMERICAN



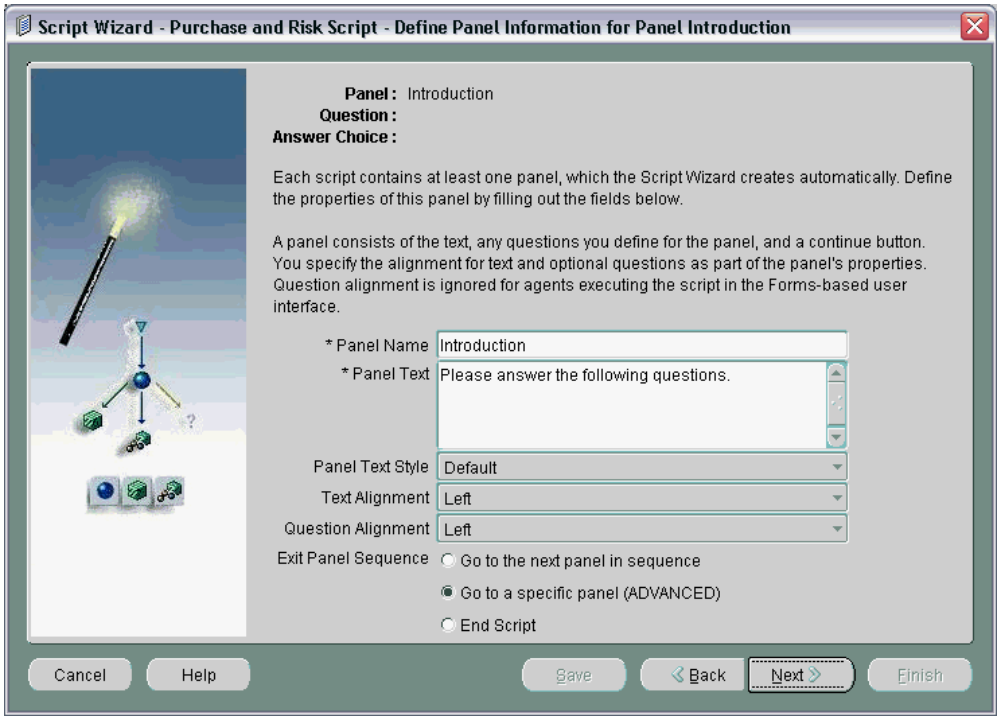
2. Click Next.

Defining the Introduction Panel

1. In the **Panel Manager,** click Create.
2. Enter the following in the **Define Panel Information** page:

Column	Value
Panel Name	Introduction
Panel Text	Please answer the following questions.
Panel Text Style	Default

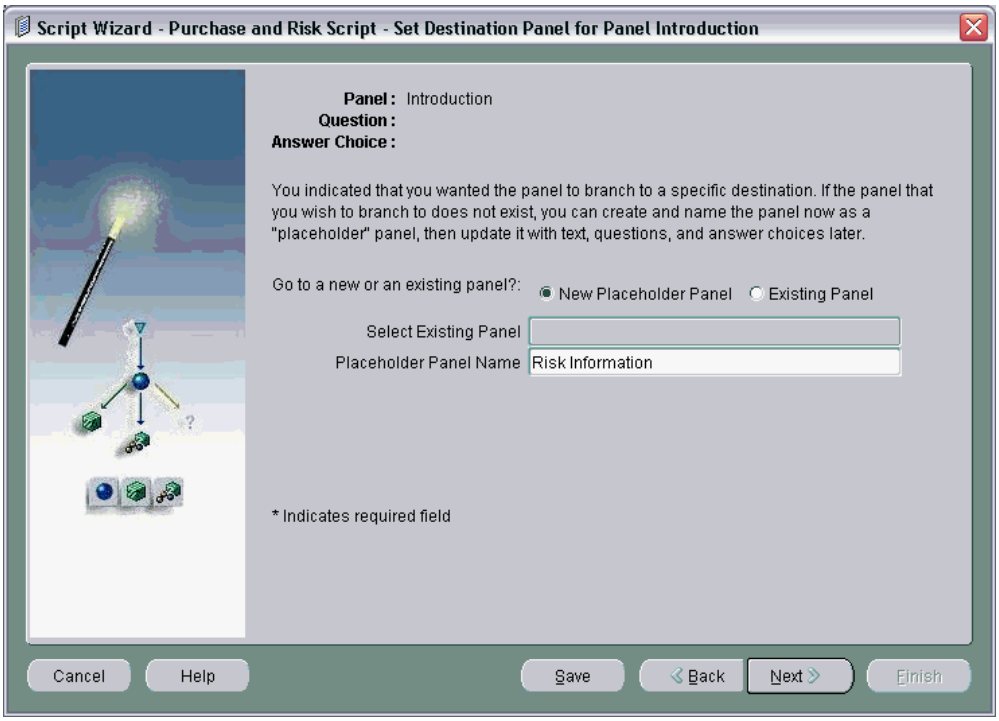
Column	Value
Text Alignment	Left
Question Alignment	Left
Exit Panel Sequence	Go to a specific panel (ADVANCED)



3. Click Next.

4. Enter the following in the **Set Destination Panel** page:

Column	Value
New Placeholder Panel	Yes
Placeholder Panel Name	Risk Information

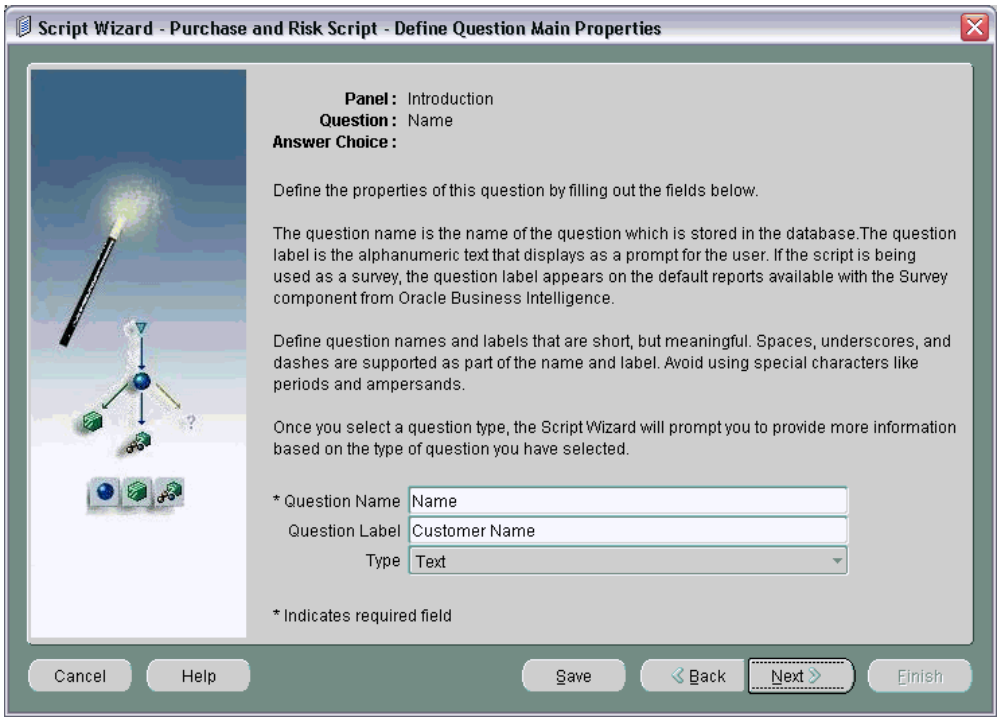


5. Click Next.

Defining the First Question of the Introduction Panel

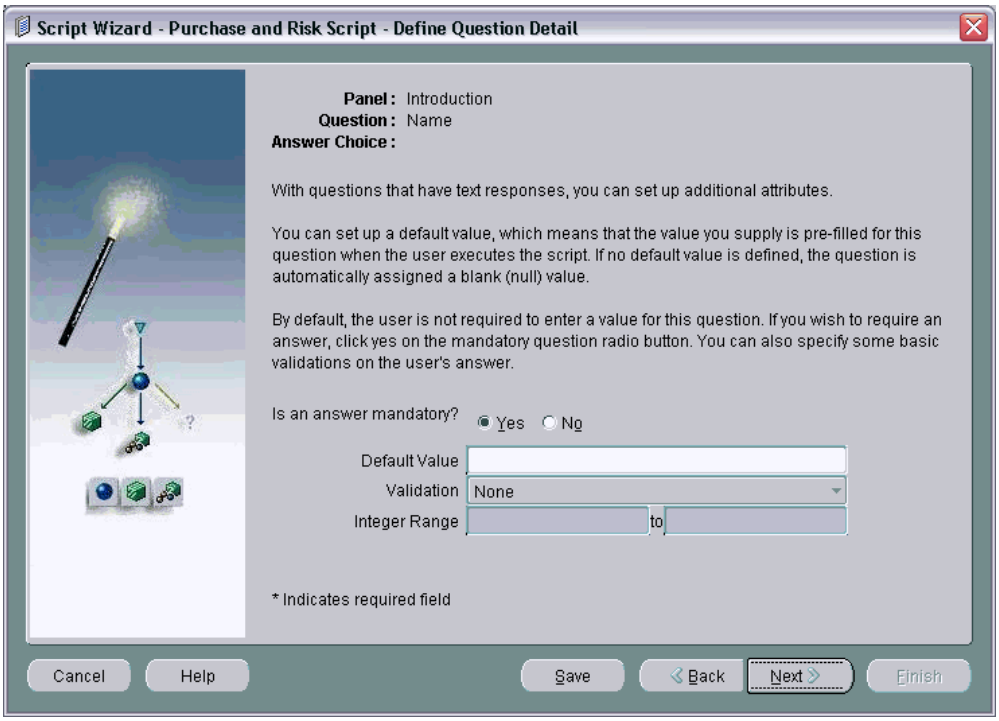
1. In the **Question Manager**, click Create.
2. Enter the following in the **Define Question Main Properties** page:

Column	Value
Question Name	Name
Question Label	Customer Name
Type	Text



3. Enter the following in the **Define Question Detail** page:

Column	Value
Is an answer mandatory?	Yes



4. Click Next.

Defining Second Question of the Introduction Panel

1. In the **Question Manager**, click Create.
2. Enter the following in the **Define Question Main Properties** page:

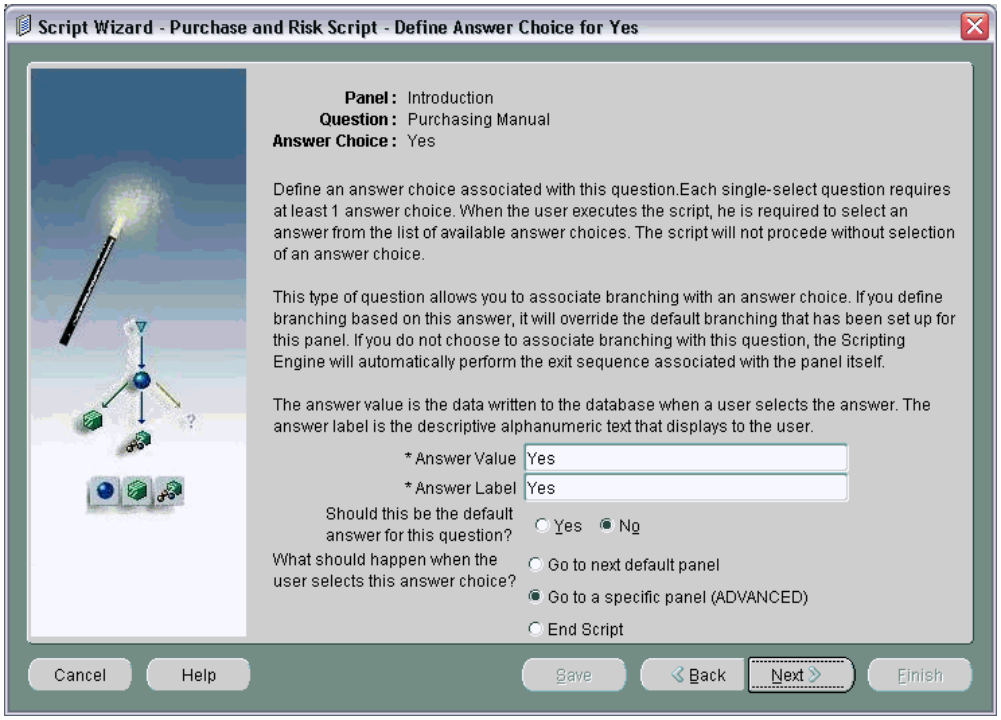
Column	Value
Question Name	Purchasing Manual
Question Text	Do you have a purchasing manual?
Type	Radio Button

3. Click Next.

Defining an Answer Choice

1. In the **Answer Manager**, click Create.
2. Enter the following in the **Define Answer Choice** page:

Column	Value
Answer Value	Yes
Answer Label	Yes
Should this be the default answer for this question?	No
What should happen when the user selects this answer choice?	Go to a specific panel (ADVANCED)



3. Enter the following in the **Set Destination Panel** page:

Column	Value
New Placeholder Panel	Yes
Placeholder Panel Name	Purchase Manual Details

4. Click Next.

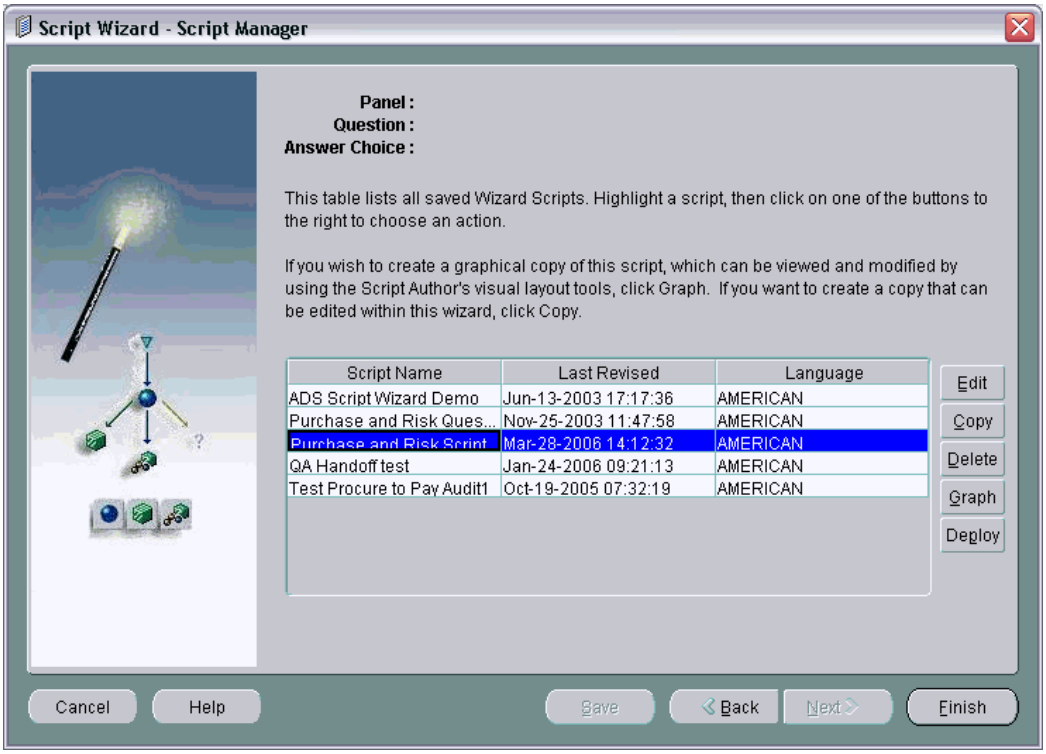
Wizard Objects in the Manager Pages

This section shows a sample of the Manager pages when all the script objects for the Purchase and Risk Script have been created.

Script Manager

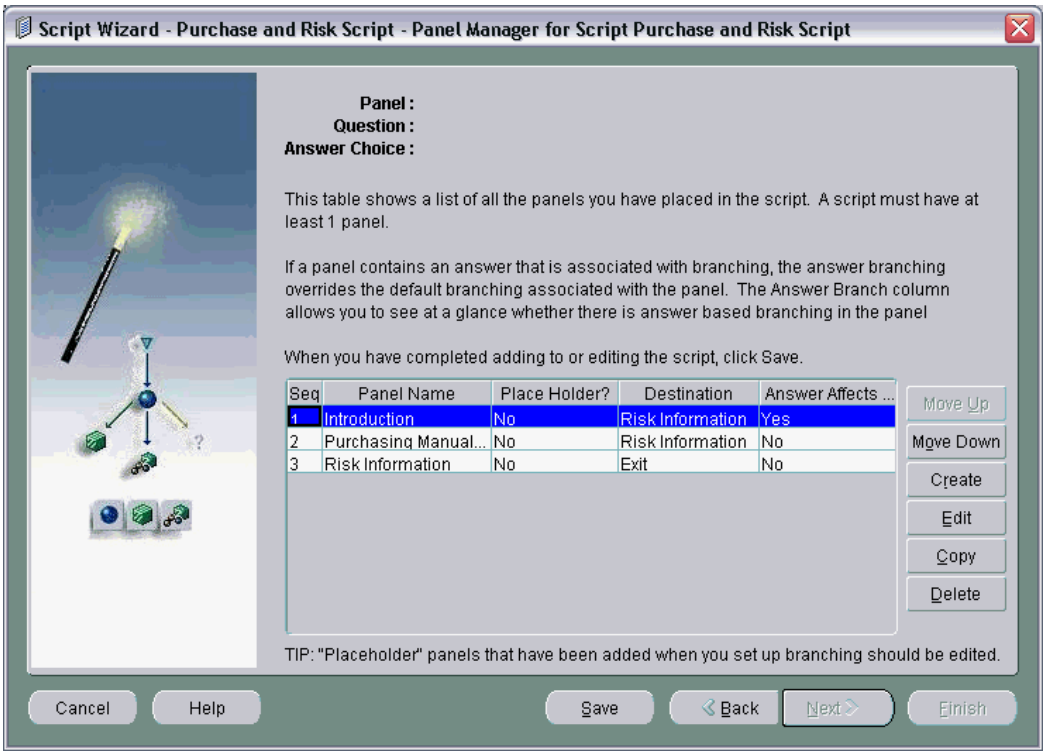
The Script Manager page that follows shows all scripts, including the Purchase and Risk Script

.



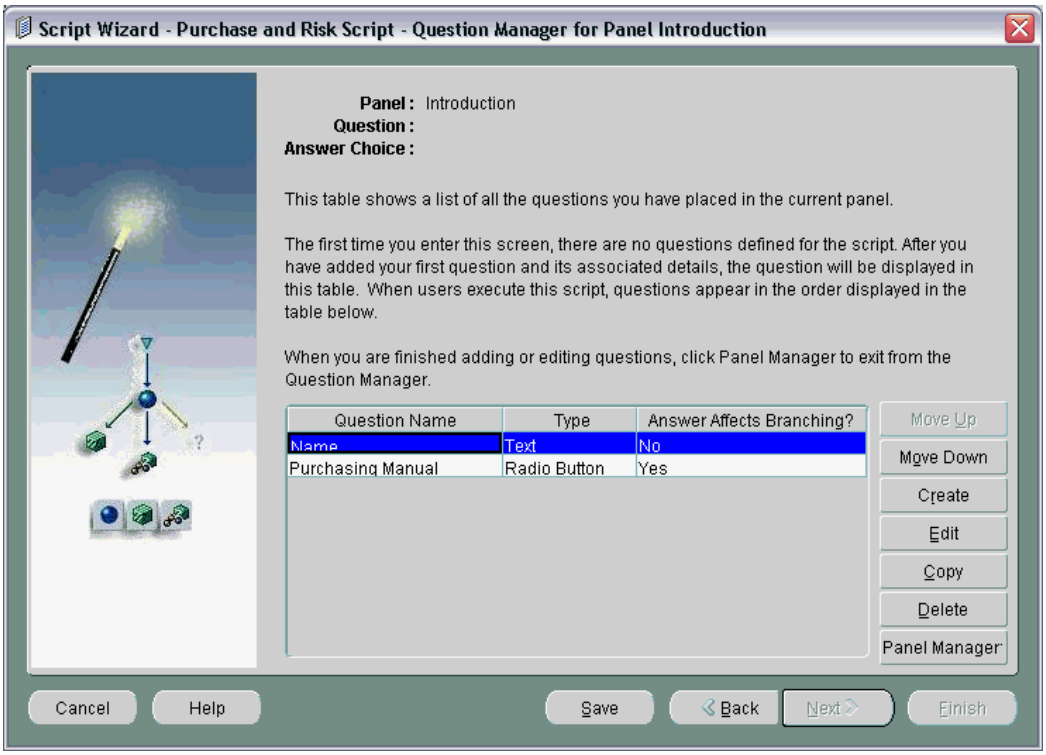
Panel Manager

The Panel Manager page that follows shows the three panels of the Purchase and Risk Script.



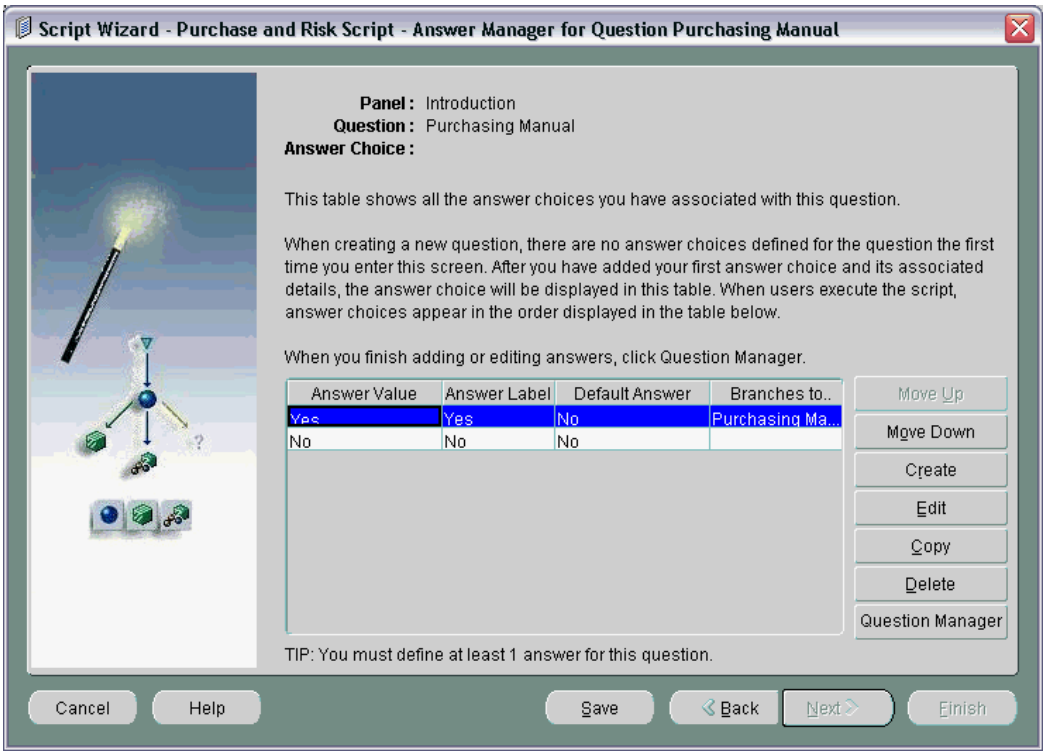
Question Manager

The Question Manager page that follows shows the questions defined for the Introduction Panel.



Answer Manager

The Answer Manager page that follows shows the answer choices defined for the answers for the Purchasing Manual question in the Introduction panel.


Script Wizard - Purchase and Risk Script - Answer Manager for Question Purchasing Manual
Panel : Introduction
Question : Purchasing Manual
Answer Choice :
This table shows all the answer choices you have associated with this question.
When creating a new question, there are no answer choices defined for the question the first time you enter this screen. After you have added your first answer choice and its associated details, the answer choice will be displayed in this table. When users execute the script, answer choices appear in the order displayed in the table below.
When you finish adding or editing answers, click Question Manager.
Answer Value
Answer Label
Default Answer
Branches to..
Yes
Yes
No
Purchasing Ma...
No
No
No
Move Up
Move Down
Create
Edit
Copy
Delete
Question Manager
TIP: You must define at least 1 answer for this question.
Cancel
Help
Save
Back
Next
Finish

Glossary

action

A Script Author command that executes at runtime upon evaluation of the object containing the action. Actions can be associated with branches, or with blocks if designated as an API block. See also pre-action and post-action.

answer collection

The recording of end user responses ("answers") to all question controls ("questions") during a script transaction. Answer collection is performed by the Scripting Engine for a script executed in either the agent interface or the Web interface, when the Answer Collection option is selected as a global script property. Answers are collected in the IES_QUESTION_DATA table in the Oracle Applications schema. Script end user responses are collected for each question designated as collectable.

answer definition

See question, page Glossary-9.

answer UI type

See question user interface type, page Glossary-9.

answer user interface type

See question user interface type, page Glossary-9.

argument

Data provided to a function or method for use in its calculations and processing. Arguments are passed to the function or method by listing them in parentheses in the function or method name. In a Java method, these are also called parameters.

best practice Java method

Oracle Applications includes Java methods written specifically for Oracle Scripting to quickly enable frequently requested script runtime functionality. Class files containing these "best practice" Java methods are included in APPS.ZIP, each in the package oracle.apps.ies.bestpractice.<Class Name>.

best practice survey

Oracle provides some pre-built survey questionnaire scripts flows to serve as examples of how to customize scripts specifically intended for use as survey questionnaires. Aspects of these scripts can also be used in scripts to be executed in the agent interface. Best practice survey scripts are placed on the APPL_TOP in the directory $IES_TOP/scripts during installation or patching of Oracle Scripting. These are provided on an as-is basis. Use of best practice surveys requires customization and is not supported.

blackboard

Virtual memory space in Oracle Scripting where information is stored in key/value pairs. This information is only retained for the duration of a single Scripting interaction.

building block

A portion of a Script Author script that contains functionality integrating with other Oracle Applications. Building blocks typically contain one or more API and typically perform a task-based operation such as creating a service request, registering for an event, and so forth. Building blocks are not intended to be used as standalone scripts, but are expected to be imported into a script and modified as appropriate. These are provided on an as-is basis. Use of building blocks requires customization and is not supported. Building block script components are created and supported by application development teams. As of this release, building block scripts are available from the Oracle Scripting and Oracle Marketing product teams. For information on building blocks, refer to *Oracle Scripting Developer's Guide* or product documentation for the specific Oracle application.

campaign

A focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose. Also see marketing campaign, page Glossary-6.

class

A set of related objects that share common characteristics. In Java, a class contains a set of methods.

collectable

A boolean property of a Script Author question within a panel. This option is selected by default in the general tab of the data dictionary of each specific question defined. Individual questions can be designated as uncollectable by clearing the collectable option. When answer collection is enabled at the global script level, all script end user responses provided in a script session will be saved for questions designated as collectable. Also see answer collection, page Glossary-1.

CRM Resource Manager

Oracle CRM Resource Manager is a responsibility included with a Rapid Install of Oracle CRM Applications so that HRMS person types (most often, employees) required by Oracle products can be created. It does not have the full functionality of HRMS. Note that if HRMS is installed, HRMS employees must be created using that tool and cannot be created with CRM Resource Manager.

cycle

A cycle is a repeatable survey event, utilizing the same script. Multiple cycles will also allow the same survey to be given through different media types, when supported. Multiple cycles are typically used to measure the same characteristics at a different moment in time.

deployed scripts

Deployed scripts are syntactically correct Script Author scripts that are executable using the Scripting Engine. The only method to deploy a script is to use Script Author (either using the **Tools > Deploy Script** command from a graphical script, or the **Save, Deploy, and Exit** command from the Script Wizard). When a script is deployed it is written to the IES_DEPLOYED_SCRIPTS table of the applications database. Deployed scripts that are selected as the survey questionnaire for a valid survey campaign can be executed in a Web browser using the Scripting Engine Web interface.

deployment

A deployment is the lowest and most granular set of requirements for a survey campaign. A deployment is associated with a specific survey and cycle, and identifies specific response start and end dates. List-based deployments identify Oracle Marketing lists and Oracle One-to-One Fulfillment templates. Deployments must be set to active status (they must be "deployed") before a survey campaign commences.

deployment ID

An identification number for each deployment in a particular instance. This is created upon setting a deployment to Active status. All respondents in a particular deployment will have the same deployment ID.

dID

See deployment ID, page Glossary-3.

error page

An error page is a survey resource that displays when the Scripting Engine encounters an error in processing a script in the Web interface (during execution of a survey or Web script in a Web browser). Survey resources defined for execution in the JTT technology stack are JSP files. Survey resources for execution in the OAF technology stack are either HTML files or (for error page or final page resources) URLs accessible to

Oracle Applications at runtime, which redirect the end user to the designated location on the Internet or intranet. Error page resources are mandatory for JTT survey campaigns. If no specific error page is defined for an OAF survey campaign, a simple error page will be automatically generated by the application.

In the JTT technology stack, scripts executed from self-service Web applications can use the seeded test error page for menu-based scripts (IESSVYMENUBASEDTESTERROR.JSP), whereas surveys or web scripts launched using the guest user may user error pages based on the seeded error page for non-hosted scripts, IESSVYTESTERROR.JSP.

final page

A final page is a survey resource that displays after the Scripting Engine processes the last panel in the flow of a script executed in the Web interface (during execution of a survey or Web script in a Web browser). Survey resources defined for execution in the JTT technology stack are JSP files. Survey resources for execution in the OAF technology stack are either HTML files or (for final page or error page resources) URLs accessible to Oracle Applications at runtime, which redirect the end user to the designated location on the Internet or intranet. When a file is used as the final page instead of a URL for redirect, it typically includes a thank-you message to the script end user, and indicates that the survey or Web script has ended successfully. Final pages often contain a hypertext link to the enterprise's home page or to the next logical destination for the end user (based on the purpose of the script). Final page resources are mandatory for JTT survey campaigns. If no specific final page is defined for an OAF survey campaign, a simple final page will be automatically generated by the application.

footer section

A footer section is a survey resource that, if associated with a survey campaign, displays at the bottom of each page (except the error page or final page) when a script is executed in a Web browser. Since each page represents a panel, the footer appears immediately below panel content (typically, below the Continue button). The footer section may reference corporate logos or other standardized graphics in GIF or JPEG format. If so, these graphics need to be stored on a directory accessible to the Web server and referenced appropriately in the footer section code. Survey resources defined for execution in the JTT technology stack are JSP files. Survey resources for execution in the OAF technology stack are HTML files. Footer section resources are the only optional survey resources for JTT survey campaigns. If no specific footer section is defined for any survey campaign (regardless of technology stack), no footer will appear on each page at runtime.

footprinting

Footprinting is the recording in the Oracle Applications database of the names of each panel in a script transaction that is visited during a script transaction, and the duration of time (in milliseconds) prior to the activation of the next panel. This feature can

provide useful script tuning data. The default for this property is **true**. When enabled (or when the Answer Collection property is **true**), this data is saved in the IES_PANEL_DATA table in the Oracle Applications schema.

graphical script

A script contains the business rules, text and graphics to progress the script end user from panel to panel at runtime. Runtime scripts are executed using any interface of the Scripting Engine component of Oracle Scripting. A graphical script is created using the graphical tools of the Script Author, the authoring and development component of Oracle Scripting. Graphical scripts can also be created by graphing a wizard script. At runtime, scripts display panels containing questions, answers, and graphics. Each panel includes a Continue button. The graphical script end user progresses through each panel. The flow of the script is controlled by business rules built using the standard Script Author graphical tools, including associated custom code, and may be a rigid flow or may change dynamically, based on the end user's responses. Graphical scripts are created, edited and deployed using the Script Author Java applet, and can be listed, deleted, or associated with custom Java from the Scripting Administration console.

header section

A header section is a survey resource that, if associated with a survey campaign, displays at the top of each page (panel) when a script is executed in a Web browser (except the error page or final page). Immediately below the header section, the content of the current panel appears. The header section may reference corporate logos or other standardized graphics in GIF or JPEG format. If so, these graphics need to be stored on a directory accessible to the Web server and referenced appropriately in the footer section code. Survey resources defined for execution in the JTT technology stack are JSP files. Survey resources for execution in the OAF technology stack are HTML files. Header section resources are mandatory for JTT survey campaigns. If no specific footer section is defined for an OAF survey campaign, no information appears above panel content for each page at runtime.

invitation

An e-mail message master document, inviting potential survey respondents (identified in an Oracle Marketing list) to participate in a survey. Each e-mail message contains a customized URL, from which the invited list member can respond to the specified survey.

JRAD

Java Rapid Application Development, an Oracle proprietary technology stack also referred to as Oracle Applications Framework (OAF). This is the technology upon which the Survey Administration console is built.

JTA

Java Technology Applications, an Oracle proprietary technology stack referring to CRM

Common Application Components.

JTF

Java Technology Framework, an Oracle proprietary technology stack referring to CRM Foundation products. Oracle applications using underlying CRM Foundation technology are further divided into the CRM Applications Foundation technology stack (known as JTA) and CRM Technology Foundation (JTT) technology stack.

JTT

Java Technology Tools, an Oracle proprietary technology stack referring to CRM Common Application Components. This is generally referred to as the Oracle CRM Technology Foundation technology stack.

list

A list is a set of records in Oracle Marketing that contains records for individuals or organizations, and contact information (at minimum, an e-mail address) for each. From an Oracle Scripting perspective, the relevance of a list is to serve as the source for potential survey respondents for a targeted survey campaign deployment. Each record represents one individual who will receive an invitation (an HTML document sent using Oracle One-to-One Fulfillment) to participate in the questionnaire. List members may also receive reminders (also HTML documents) to participate in a survey, which includes the survey end date. Lists are built using Oracle Marketing's list management feature. Use of targeted (list-based) surveys requires Oracle Marketing and Oracle One-to-One Fulfillment.

list member

Any individual included in a list built using Oracle Marketing's list management feature. All list members are potential survey respondents.

marketing campaign

A marketing campaign is a collection of requirements created in Oracle Marketing for the purpose of executing specific business rules of a marketing effort within an enterprise or interaction center. It is the primary organizational unit of a focused marketing effort. Campaigns define the marketing effort and, through association with other units, define schedules for execution and target populations for the effort. One marketing campaign must have at least one campaign schedule associated with it, and may have many. Campaign schedules can be targeted for different execution channels (such as e-mail, telephony, or traditional postal service mail). Each campaign schedule has a target group (list or lists created using Oracle Marketing to define the target population for the campaign). The marketing campaign is used by several Oracle Applications, including Oracle TeleSales, Oracle Collections, and Oracle Advanced Outbound Telephony.

master document

A master document is a message template used in Oracle One-to-One Fulfillment. Master documents typically include merge fields populated by an associated SQL query. Upon execution of the query, a copy of the master document is created for each list member, containing the data returned by the query. This creates a personalized message. From a survey perspective, master documents are invitation or reminder messages in HTML format that are sent by e-mail from the fulfillment engine to members of a defined Oracle Marketing **list.** Master documents are required for targeted survey deployment operations. Master documents and their associated queries are associated with specific Oracle One-to-One Fulfillment templates.

merge field

Merge fields are data elements in an Oracle One-to-One Fulfillment master document surrounded by open and close guillemet (for example, <<merge_field>>). When a fulfillment request is executed, specified table columns or views are queried. For each list member record, the master document is replicated, with the actual data returned from the query embedded or merged in the master document replicate in place of the merge field. List-based surveys require at minimum one merge field (the survey URL that the recipient must access to participate in the survey). Merge fields must correspond to data returned from a query.

method

A task that can be performed with specific data items or properties. A Java method can be a function (resulting in a return value) or a procedure.

Oracle HRMS

Oracle HRMS is a full-featured enterprise human resources management system. It is an ERP application using Oracle Forms technology to manage human resources used by a variety of ERP and CRM applications.

panel

A panel is the primary object of a script, which contains the information that is displayed by the Scripting Engine (in the panel presentation area or browser window) at runtime. Each panel must contain at least one question and may contain as many as needed. Each question in a panel displays a corresponding question user interface control (button, checkbox, text area, drop-down, and so forth) with which the script end user enters a response at runtime. Panels are created and edited using the Script Wizard or graphical tools.

Panels in wizard or graphical scripts may contain static panel text, and one or more questions. Questions of the appropriate type that are explicitly defined in a wizard script may include question-level validation commands. In addition, panels in graphical scripts may contain graphic images, dynamic text using embedded values, and validation, database or cursor lookup, or other commands (at the panel level or the

question level).

If the purpose of a wizard script panel is simply to provide information to the end user, then it can automatically generate a question called Continue, which displays a Continue button at runtime. Panels created with graphical tools must contain one or more questions, for which the properties of each must be explicitly defined.

Whether built with the Script Wizard or in a graphical script, a panel is the only Script Author object to display at runtime in both interfaces (groups, blocks and branches simply contain information for processing a script).

PHP

Personal home page. Accessed when logging into Oracle Self Service applications, the personal home page lists the responsibilities available to the logged in Oracle Applications account in two categories: Self Service, to access self-service applications, and Applications, for access to other Oracle applications.

post-action

A Script Author command that executes immediately following the evaluation of the object with which the post-action is associated, and prior to evaluation of the next object in the flow of the script. Post-actions can be associated with panels, blocks, groups, or the global script. See also action and pre-action.

pre-action

A Script Author command that executes prior to the evaluation of the object with which the pre-action is associated. Pre-actions can be associated with panels, blocks, groups, or the global script. See also action and post-action.

prototype

Prototype is a boolean characteristic of survey campaigns. Prototype survey campaigns are identical to standard survey campaigns, except that the script used as the survey campaign questionnaire is not locked. The purpose is to allow survey campaign administrators more freedom to refine requirements for survey campaigns, including modification to the script for the designated survey campaign. You can exclude prototype survey campaigns from survey campaign lists by selecting the **Exclude prototypes** option. This characteristic can only be selected or cleared while a survey campaign status is Open.

published scripts

Published scripts are stored in the IES_DEV_SCRIPTS table of the applications database, and are not executable. All wizard scripts are published scripts. To publish a graphical script you must be explicitly save it to the database from Script Author by selecting **File > Save As** and selecting the Database tab. Published wizard scripts are always syntactically valid; graphical scripts can be published as works in progress, regardless of their contents.

query

A specific statement of request for information to be returned in the form of records from the database. Queries are constructed in structured query language (SQL). For targeted survey deployments, each Oracle One-to-One Fulfillment master document must have an associated query which obtains information populated into the resulting document in placeholders called merge fields. A query must be valid to return any of the requested information.

question

In Script Author, a question is the property of a panel which results in the appearance of a question user interface (UI) control in the panel at runtime. All panels must contain at least one question and may contain any number. Questions may have commands associated with them in Script Author to validate responses at runtime, or to provide default user responses. When viewed from a graphical script, each question has a data dictionary associated with it, in which information (answer choices, commands) is stored.

question control

See question user interface type, page Glossary-9.

question UI type

See question user interface type, page Glossary-9.

question user interface type

The choice of user interface (UI) type selected by the script developer for a question in a panel determines the kind of control that displays at runtime with which the end user can provide input. The full range of question UI types supported by Oracle Scripting include text fields, text areas, radio buttons, checkboxes, buttons, drop-down lists, password fields, checkbox groups and multi-select list boxes. All nine are available in graphical scripts. Wizard scripts do not support the single check box or button question UI types. Each question supports only one question UI type. Once selected in a graphical script, the question user interface type cannot be changed (you must delete the entire question control and add a new one). Panels in graphical scripts that require the button UI type cannot contain any other questions. Some question UI types allow null or open-ended input (text fields, text areas, password fields, checkbox, checkbox group, multi-select list boxes). Others require end user input at runtime (radio buttons, buttons, drop-down lists) unless a default answer choice is associated with that question.

reminder

An e-mail message master document, reminding potential survey respondents (already invited to take a survey) of the appropriate URL to access to respond to the survey. Reminders often include information such as the last day the recipient may participate

in the survey. Using the same master document, reminders can be sent in a batch process on a predefined schedule, or can be sent manually for any individual list member.

respondent

An individual responding to a request for information (in the form of participating in a survey questionnaire or Web script). For targeted survey deployments, respondents are list members who respond to an invitation sent by e-mail to participate in the survey.

respondent ID

A unique identification number for each list-based respondent in a particular instance. The URL for a list member includes as parameters both the deployment ID common to all participants in a given deployment and each list member's unique respondent ID.

rID

See respondent ID, page Glossary-10.

responsibility

A responsibility is a level of authority that allows a user to access specific functionality and data in Oracle Applications. Responsibilities are assigned by a user with the System Administrator responsibility.

root graph

The root graph is the first graph that appears when a graphical script is opened. Processing for any script begins and ends on the root graph. For a script with no groups or blocks, the root graph is the only graph contained in the script. When branching logic is employed in a script which directs the script to subgraphs of the root graph, each subgraph must be terminated, returning processing to its higher level graph. The root graph is the highest level graph in a script.

runtime

Runtime is the execution of any script created using Script Author. Scripts are executed using the Scripting Engine component, which has two interfaces - the agent interface and the Web interface.

The agent interface executes as a Java window launched from an Oracle business application form; it includes the panel presentation area, a script information area, a shortcut button area, and a progress area, wrapped in a frame with a Disconnect button and, optionally, a Suspend button.

The Web interface executes a script in a Web browser. The Web interface displays the panel presentation area, and the following buttons: Back, Next, Reset to Default, and optionally Save for Later.

Before any script can be executed in the Web interface, survey administration is

required, including setting up resources, creating a hierarchical set of objects - a survey campaign, cycle, and deployment - and activating the deployment.

sample

The population from which information is solicited by survey.

script

A script is a miniature program built using the Script Author component of Oracle Scripting and executed at runtime using the Scripting Engine component of Oracle Scripting. The purpose of a script is to facilitate the flow of information between an enterprise and other parties. Each script enforces business rules programmed into it by script developers. Scripts consist primarily of panels with at least one question each; at runtime, script end users must click the Continue button in each panel to progress through the script. Graphical scripts also may contain groups and blocks. By design, scripts either enforce a rigid flow, or employ carefully constructed logic to route the end user through different potential paths in a script. to appropriate panels, based on responses to earlier script questions. Script Author provides the ability to create two kinds of scripts: wizard scripts, page 2-2 and graphical scripts, page Glossary-5. Regardless of creation method, after deployment a script can be executed in any interface of the Scripting Engine. Executing a script in a Web browser requires the creation and activation of a survey campaign deployment.

A script contains the business rules, text and graphics to progress the script end user from panel to panel. Scripts are created using the Script Author component of Oracle Scripting. At runtime, scripts are executed using any interface of the Scripting Engine component of Oracle Scripting. The same script can be executed in the agent interface, or (after creating and deploying a survey campaign) executed in a Web browser as a survey questionnaire or web script accessed from a self-service Web application. Scripts are further qualified by type. See wizard scripts, page 2-2 and graphical scripts, page Glossary-5.

script information area

The script information area is a programmable feature of the Script Author in which script developers can place information that appears as a header in any script executed in the Scripting Engine agent interface at runtime. The script information area is a global script property and appears in every session of a script executed in the agent interface when this information is defined. It is referred to in the Script Author user interface as the Static Panel. You can access this area by selecting **File > Script Properties > Static Panel** or by right-clicking on an empty area of the canvas and selecting **Edit Blob Properties > Static Panel**.

static panel

See script information area, page Glossary-11.

survey campaign

A survey campaign is the collection of requirements needed to execute a script in a Web browser using the Scripting Engine Web interface. This is the super class that references the global script. A survey campaign must have information for at least one cycle and at least one deployment. Survey campaigns are created using the Survey Administration console. As prerequisites to creating a survey campaign, you must create and deploy from Script Author the script containing questions and answer choices, and you must define survey resources.

survey questionnaire

Set of questions built using the Script Author component of Oracle Scripting for execution in the Scripting Engine as a survey questionnaire, typically to gather feedback, obtain market data, tabulate opinion, measure satisfaction, or otherwise collect data directly from specific (customers, employees, prospects) or non-specific target populations. A survey questionnaire script is physically identical to a script executed in the agent interface, although it may be customized to take advantage of sophisticated HTML interpretation available to Web browsers.

survey resource

Survey resources provide functionality to scripts executing in the Scripting Engine Web interface. Survey resource definition is accomplished from the Survey Administration console **Survey Resources** tab >**Create**, and must be defined before they can be used in a survey campaign. The four survey resource types include header section, footer section, error page, and final page resources. Header and footer sections appear in each page of the Web browser, just above or below panel contents, respectively. Error page resources are displayed when an error condition occurs during script execution in a Web browser. Final page resources are displayed after the Scripting Engine processes the last panel in the flow of a script. For survey campaigns defined in the JTT technology stack, survey resources are JSP files, and all resources except the footer page are mandatory. For survey campaigns defined in the OAF technology stack, survey resources are HTML files. For OAF survey campaigns, error page or final page resources can also be URLs accessible to the Web server at script runtime, which redirect the script end user to the specified URL during an error or upon completing the script, respectively.

Physical JSP survey resources corresponding to survey resource definitions must be uploaded manually into the $OA_HTML directory on the applications server and are served by the Web server as appropriate. For OAF survey resources, the HTML files corresponding to the survey resource definition are uploaded to the database at the time of survey resource definition.

For JTT survey campaigns, you can use seeded JSP test resources to test your implementation, or you can customize your own resources based on these seeded resources. These seeded JavaServer page files (IESSVYTESTHEADER.JSP, IESSVYTESTTHANKU.JSP, IESSVYTESTERROR.JSP and

IESSVYMENUBASEDERROR.JSP) are located in $OA_HTML in your environment. For error pages with a hosting type of menu-based, use the appropriate seeded resource (IESSVYMENUBASEDERROR.JSP.

survey respondent

See respondent, page Glossary-10.

SYSDATE

This refers to the current Systems Date of the database. When creating records in a database the SYSDATE is typically time-stamped as creation date, and SYSDATE is also used as a default setting for creating responsibilities from an Oracle Applications perspective.

template

Oracle One-to-One Fulfillment's term for a group of master documents and any collateral. Templates may also include campaign specifications. Templates are identified with a survey campaign at the deployment level to identify invitation and reminder master documents and their associated queries. Only required for targeted (list-based) survey deployments. No collateral is typically associated with a survey campaign.

trading community architecture (TCA)

The trading community architecture is a customer model designed to support complex trading communities. The entire Oracle E-Business Suite uses the TCA customer model, providing a common model for customer data in all Oracle business applications. TCA strives to model *all* relationships within a trading community, supporting business-to-business (B2B) *and* business-to-consumer (B2C) models equally.

Uniform Resource Locator (URL)

A specific string of characters that identifies resources located on the Web. Also known as Uniform Resource Identifiers (URIs), the provide navigation to documents, images, downloadable files, services, electronic mailboxes, and other resources to Web browser users with one click. Full URLs include access methods or protocols such as HTTP or FTP, generally proceeded by a colon and two slashes, and then the full location. Nested directory levels are represented by additional slashes.

Web script

A Script Author script executed in a Web browser from an Oracle self-service Web application such as Oracle iSupport or Oracle iStore is referred to as a Web script. Like all scripts executed in the Scripting Engine Web interface, this requires as a prerequisite an active survey deployment. The URL for this deployment is then embedded in the self-service application as a hyperlink. When the end user selects the link, a new Web browser window opens, the user's current Oracle applications validation information is used to authenticate the script session, and the script executes in the new browser

window. Web scripts are a valuable resource to provide scripted information to Web application users or to solicit feedback to the enterprise regarding the application user's experience.

wizard script

A wizard script is a script created using the Script Wizard feature of the Script Author, the authoring and development component of Oracle Scripting. Like graphical scripts, wizard scripts are executed in any Scripting Engine interface. At runtime, they display panels containing questions, answers, and graphics. Each panel includes a Continue button required to be clicked by the script end user at runtime to progress through each panel. The flow of the script is controlled by business rules built using the Script Wizard. Based on the choices of the script developer, the end user's flow through the script may be rigid (the same panels are seen regardless of how the user answers) or may change dynamically (demonstrating different flows based on the end user's responses). Wizard scripts can be listed, created, copied, edited, deployed, or converted to a graphical script only from the Script Wizard feature of the Script Author.