Oracle® Scripting

User Guide

Release 11*i*

Part No. B10207-02

August 2004

ORACLE®

Oracle Scripting User Guide, Release 11*i*

Part No. B10207-02

Primary Author: Samuel Besalel

Contributors: Stephanie Smith, Norman Chan, Elizabeth Chambers-Carbone, Kumar Pandey, Rathna Sundaralingam, Pradeep Kotha.

Contents

2 Planning Oracle Scripting Projects

3 Using the Script Wizard

4 Using Script Author

Send Us Your Comments

Oracle Scripting User Guide, Release 11*i*

Part No. B10207-02

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

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

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

- Electronic mail: appsdoc_us@oracle.com
- FAX: (650) 506-7200 Attn: Oracle Applications Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle Applications Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

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

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

Preface

Audience for This Guide

Welcome to Release 11*i* of the Oracle® Scripting User's Guide.

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

- The principles and customary practices of your business area.
- Oracle Scripting

  If you have never used Oracle Scripting, Oracle suggests you attend one or more of the Oracle Scripting training classes available through Oracle University.
- The Oracle Applications graphical user interface.

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

See Other Information Sources for more information about Oracle Applications product information.

How To Use This Guide

This guide contains the information you need to understand and use Oracle Scripting.

- **Understanding Oracle Scripting** includes an overview of Oracle Scripting, followed by sections to help you understand each of the four components of Oracle Scripting: Understanding Script Author, Understanding the Scripting Engine, Understanding the Scripting Administration Console, and Understanding the Survey Administration Console.

- **Planning Oracle Scripting Projects** describes planning aspects required for Oracle Scripting, including planning Scripting Engine projects, and planning Oracle Scripting survey campaigns.
- **Using the Script Wizard** details the mechanics of creating a script using a simple wizard interface introduced to Script Author in release 11.5.9 (Interaction Center Family Pack Q). Learn to create and deploy scripts, re-use questions and answer choices, reduce keystrokes and data entry errors, and reduce some repetitious work needed to create and modify a script.
- **Using Script Author** provides, in great detail, task-based procedures to create graphical scripts using Script Author.
- **Using the Scripting Engine** explores Scripting Engine concepts, and describes the two Scripting Engine interfaces. This section briefly describes how users of either interface launch and execute scripts. Tasks specific to the agent interface, such as launching, suspending, or resuming a script in the Agent Interface, are also detailed.

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. 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 additional information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, 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, JAWS 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.

Other Information Sources

You can choose from many sources of information, including online documentation, training, and support services, to increase your knowledge and understanding of Oracle Scripting.

If this guide refers you to other Oracle Applications documentation, use only the Release 11*i* versions of those guides.

Online Documentation

All Oracle Applications documentation is available online (HTML or PDF). Some online help patches are available on Oracle*MetaLink* or Oracle iSupport. Oracle Scripting online help is available for the Scripting Administration and Survey Administration components.

Related Documentation

Oracle Scripting shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other product documentation when you set up and use Oracle Scripting.

You can read the documents online by choosing Library from the expandable menu on your HTML help window, by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle Store at http://oraclestore.oracle.com.

Documents Related to All Products

Oracle Applications User's Guide

This guide explains how to enter data, query, run reports, and navigate using the graphical user interface (GUI) available with this release of Oracle Scripting (and any other Oracle Applications products). This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

You can access this user's guide online by choosing "Getting Started with Oracle Applications" from any Oracle Applications help file.

Documents Related to This Product

Oracle Scripting Developer's Guide 11i

This guide is intended for advanced users of Script Author. Included is information on the various Script Author commands, how to use them and where to define them; a description of Best Practice Java methods you can use to extend your scripts; and detailed information on performing advanced tasks in a graphical script. This document lists seeded reusable commands, instructing on their use and integration with seeded building block scripts; and describes each building block script. Information on Best Practice survey scripts is also included. Finally, extensive information is included regarding customizing scripts, with particular attention to customizing panel layouts.

Oracle Scripting Implementation Guide 11*i*

This guide describes how to implement Oracle Scripting components and test the implementation appropriately. This guide also details task-based steps for administering Oracle Scripting. You can access Oracle Scripting administration procedures online by choosing "Getting Started with Oracle Applications" from any Oracle Applications help file.

Oracle Scripting User Guide 11i

This guide explains how to understand the various components of Oracle Scripting. It includes information on planning Oracle Scripting implementations, details specifics on using Script Author to create graphical and wizard scripts, and describes how to execute scripts using the Scripting Engine (in both the agent and Web interfaces).

Installation and System Administration

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind Applications-wide features such as Business Intelligence (BIS), languages and character sets, and Self-Service Web Applications.

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled

using Oracle Rapid Install, which minimizes the time to install Oracle Applications, the Oracle8 technology stack, and the Oracle8*i* Server technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user's guides and implementation guides.

Oracle Applications Supplemental CRM Installation Steps

This guide contains specific steps needed to complete installation of a few of the CRM products. The steps should be done immediately following that tasks given in the Installing Oracle Applications guide.

Upgrading Oracle Applications

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process and lists database and product-specific upgrade tasks. You must be either at Release 10.7 (NCA, SmartClient, or character mode) or Release 11.0, to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

Maintaining Oracle Applications

Use this guide to help you run the various AD utilities, such as AutoUpgrade, AutoPatch, AD Administration, AD Controller, AD Relink, License Manager, and others. It contains how-to steps, screenshots, and other information that you need to run the AD utilities. This guide also provides information on maintaining the Oracle applications file system and database.

Oracle Applications System Administrator's Guide

This guide provides planning and reference information for the Oracle Applications System Administrator. It contains information on how to define security, customize menus and online help, and manage concurrent processing.

Oracle Alert User's Guide

This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.

Oracle Applications Developer's Guide

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle*

Applications User Interface Standards for Forms-Based Products. It also provides information to help you build your custom Oracle Forms Developer 6*i* forms so that they integrate with Oracle Applications.

Oracle Applications User Interface Standards for Forms-Based Products

This guide contains the user interface (UI) standards followed by the Oracle Applications development staff. It describes the UI for the Oracle Applications products and how to apply this UI to the design of an application built by using Oracle Forms.

Other Implementation Documentation

Multiple Reporting Currencies in Oracle Applications

If you use the Multiple Reporting Currencies feature to record transactions in more than one currency, use this manual before implementing Oracle Scripting. This manual details additional steps and setup considerations for implementing Oracle Scripting with this feature.

Multiple Organizations in Oracle Applications

This guide describes how to set up and use Oracle Scripting with Oracle Applications' Multiple Organization support feature, so you can define and support different organization structures when running a single installation of Oracle Scripting.

Oracle Workflow Guide

This guide explains how to define new workflow business processes as well as customize existing Oracle Applications-embedded workflow processes.You also use this guide to complete the setup steps necessary for any Oracle Applications product that includes workflow-enabled processes.

Oracle Applications Flexfields Guide

This guide provides flexfields planning, setup and reference information for the Oracle Scripting implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications product data. This manual also provides information on creating custom reports on flexfields data.

Oracle eTechnical Reference Manuals

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific

Oracle Applications product. This information helps you convert data from your existing applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on Oracle*MetaLink*

Oracle Manufacturing APIs and Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes APIs and open interfaces found in Oracle Manufacturing.

Oracle Order Management Suite APIs and Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes API's and open interfaces found in Oracle Order Management Suite.

Oracle Applications Message Reference Manual

This manual describes Oracle Applications messages. This manual is available in HTML format on the documentation CD-ROM for Release 11i.

Oracle CRM Common Application Components Implementation Guide

Many CRM products use components from CRM Common Application Components (formerly referred to as the CRM Application Foundation). Use this guide to correctly implement CRM Common Application Components.

Training and Support

Training

Oracle offers training courses to help you and your staff master Oracle Scripting and reach full productivity quickly. You have a choice of educational environments. You can attend courses offered by Oracle University at any one of our many Education Centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization structure, terminology, and data as examples in a customized training session delivered at your own facility.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Scripting working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle8*i* server, and your hardware and software environment.

Oracle*MetaLink*

Oracle*MetaLink* is your self-service support connection with web, telephone menu, and e-mail alternatives. Oracle supplies these technologies for your convenience, available 24 hours a day, 7 days a week. With Oracle*MetaLink*, you can obtain information and advice from technical libraries and forums, download patches, download the latest documentation, look at bug details, and create or update TARs. To use Oracle*MetaLink*, register at (http://metalink.oracle.com).

Alerts: You should check Oracle*MetaLink* alerts before you begin to install or upgrade any of your Oracle Applications. Navigate to the Alerts page as follows: Technical Libraries/ERP Applications/Applications Installation and Upgrade/Alerts.

Self-Service Toolkit: You may also find information by navigating to the Self-Service Toolkit page as follows: Technical Libraries/ERP Applications/Applications Installation and Upgrade.

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 Oracle Applications 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.

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 160 software modules for financial management, supply chain management, manufacturing, project systems, human resources and customer relationship management.

Oracle products are available for mainframes, minicomputers, personal computers, network computers and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 145 countries around the world.

1

Understanding Oracle Scripting

This section includes the following topics:

1.1 Oracle Scripting Overview

This section includes the following topics:

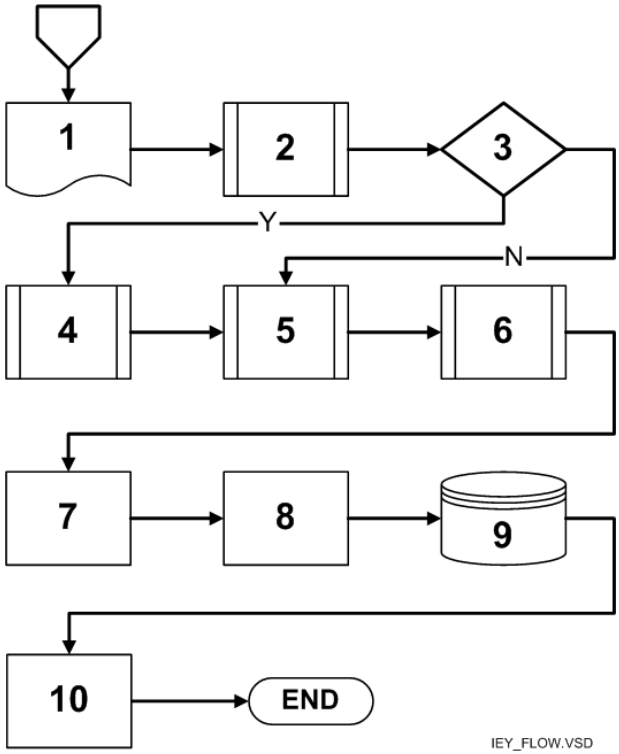
See Also

- Understanding Script Author
- Understanding the Scripting Engine
- Understanding the Scripting Administration Console
- Understanding the Survey Administration Console

1.1.1 What Is Oracle Scripting?

Oracle Scripting is a set of tools to facilitate the process of gathering or exchanging information for the benefit of the enterprise. To accomplish this goal, Oracle Scripting uses four components.

Figure 1–1 Four Components of Oracle Scripting



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 any Scripting Engine interface. Each implementation of Oracle Scripting employs at least one customized script developed by Oracle Consulting, consulting partners, or the enterprise.
2. Using the Scripting Engine component, script end users execute the script in one of two runtime interfaces.

Using the *Scripting Engine agent interface*, scripts execute as a series of Java components wrapped in an Oracle form. Users of the agent interface are interaction center agents; this interface is used in combination with Oracle business applications.

Using the *Scripting Engine Web interface*, scripts execute in any Oracle Applications 11*i*-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. Any script executed in the Web interface requires survey campaign administration.

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, survey campaigns and associated survey resources are administered. Any script executed in the Scripting Engine Web interface requires a survey campaign to be defined and one of its deployments activated. Survey resources can include header and footer sections, error pages and final pages. For each script executed in the Web interface, the appropriate resources display.

See Also

- Why Use Oracle Scripting?
- Why Is Oracle Scripting in the Interaction Center Suite?

1.1.2 Why Use Oracle Scripting?

This section includes the following topics:

- Applications for Each Component
- Applications for a Script
- Control Conversation Flow

See Also

- What Is Oracle Scripting?
- Why Is Oracle Scripting in the Interaction Center Suite?

1.1.2.1 Applications for Each Component

The following summarizes the purpose for using 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).

See Also

- Applications for a Script
- Control Conversation Flow

1.1.2.2 Applications for a Script

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

- Scripts executed in the Scripting Engine agent interface can assist enterprise agents in gathering data from or providing information to customers and prospects.

  The Scripting Engine agent interface can be used in inbound, outbound, or blended interaction centers in order to guide agents through their interactions with customers or prospects. This can reduce agent training time, ensure consistent corporate messages are communicated to customers or prospects, and enforce rigid flow of data or communicate legal information required by an enterprise's legal division.

  Scripts in the Scripting Engine agent interface can be launched from Oracle TeleSales, Oracle Collections, and the Customer Support component of Oracle TeleService. Oracle Corporation recommends using Oracle Scripting in the agent interface in combination with one of these integrated business applications.

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

- A script can serve to unify a sales, service, or collections agent's desktop by integrating aspects of various applications.
- 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.
  - Leveraging other Oracle Applications (Oracle One-to-One Fulfillment and Oracle Marketing Online), members of a defined population or sample (represented as an Oracle Marketing list) can be sent e-mail invitations or reminders to participate in a survey questionnaire. The list includes a link to an active survey deployment.

  Any script executed in the Scripting Engine Web interface (as a survey questionnaire or as a Web script) requires survey campaign administration.

  Scripts executed in the Web interface can enable an enterprise to poll a target audience, receive and analyze the responses, and take action based on the compiled survey data. Typical uses include customer satisfaction surveys, customer interest surveys, new product concept surveys, vendor surveys, etc. Audiences include current and prospective customers, consumers using competing products or services, existing customer support (iSupport) customers, and narrowly targeted populations such as the employees within a company, shareholders of stock, and so forth.

See Also

- Applications for Each Component
- Control Conversation Flow

1.1.2.3 Control Conversation Flow

The information displayed to a script end user is based on a predetermined set of requirements that have been programmed into a script. Scripts have two possibilities for end user flow, based on the business rules built into the script and the requirements of the enterprise:

1. **Rigid flow.** A script may enforce consistency and flow, displaying the same set of information to all end users for each script session.

   This requirement is typical of survey questionnaires, or enterprises that must meet strict legal or other stringently enforced established regulations.

2. **Dynamic flow.** Each end user's progression through the script (each script session) may change the flow dynamically, based on responses provided by the end user, custom code, and preprogrammed events or conditions (for example, date, time of day, presence of customer information in the database, etc.).

   Scripts with branching logic and dynamic flow are more efficient, and correspondingly more complex to plan and build. The requirement for a branching script is typical of the following applications: voluntary Web-based surveys, multi-purpose scripts, interaction center scripts for blended (inbound and outbound) environments, or scripts tuned to reduce average agent talk time.

The determination as to whether a script will employ rigid or dynamic flow is an important planning factor for individuals in an enterprise responsible for determining script business requirements. Flow requirements must be carefully understood and communicated to script developers, who translate business requirements and business rules into a script that meets the enterprise requirements.

See Also

- Applications for Each Component
- Applications for a Script

1.1.3 Why Is Oracle Scripting in the Interaction Center Suite?

The Scripting Engine agent interface provides enterprises with a method of scripting interactions with customers or prospects and integrating desktop workflow between various applications. It is intended to be used in combination with integrated Oracle business applications (Oracle TeleService, Oracle TeleSales and Oracle Collections) to take full advantage of the contact handling and sales or service features of those applications, respectively. Interaction centers leveraging

business applications and the Scripting Engine therefore have the full benefit of interaction center communications.

Both of the Scripting Engine interfaces (the agent interface and the Web interface) allow an enterprise to capture customer needs, perform market research, and measure customer satisfaction. Additionally, Oracle Scripting's survey functionality utilizing targeted survey deployments provides enterprises with a powerful information gathering tool to gather data from targeted populations. Information gained as a result of executing a survey campaign can be used to drive new business initiatives, immediately spot customer satisfaction issues, test new products or ideas, and provide baseline measures for satisfaction improvement programs. These help to provide a three hundred and sixty-degree view of an enterprise's customers, which is part of the critical mission of customer relationship management applications.

See Also

- What Is Oracle Scripting?
- Why Use Oracle Scripting?

1.2 Understanding Script Author

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.

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.

This section includes the following topics:

- Script Author Terminology
- Script Author Concepts
- Script Author Features

See Also

- Oracle Scripting Overview
- Understanding the Scripting Engine
- Understanding the Scripting Administration Console
- Understanding the Survey Administration Console

1.2.1 Script Author Terminology

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

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 employ a visual paradigm similar to a flowchart. Using drag and drop techniques, trained 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). 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.

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 objects and branch types accessible to the graphical script user interface. 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.

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 series of Java components wrapped in an Oracle Form (including 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 (only the panel presentation area is displayed). 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.

See Also

- Script Author Concepts
- Script Author Features

1.2.2 Script Author Concepts

This section includes the following topics:

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users

- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

See Also

- Script Author Terminology
- Script Author Features

1.2.2.1 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).

When displayed at runtime, each panel includes a Continue button. At runtime, script end users must click the Continue button to progress through the script.

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: wizard scripts and graphical scripts. Regardless of creation method, a script can be executed in any interface of the Scripting Engine.

See Also

- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.2 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).

Panels are the only script objects to display at runtime. Each contains at least one question user interface control, 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. Graphical scripts allow the full range of branch types: default branches, distinct branches, conditional branches and indeterminate branches. Each has rules it follows to control flow of the script at runtime.

In a graphical script, you have access to functionality not available to wizard scripts. For example:

- Graphical scripts provide users with the ability to customize a panel's visual appearance by allowing you to export, modify, and reimport panel HTML.
- Graphical scripts intended for execution in the Scripting Engine agent interface may also 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.

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.

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 Script Author 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

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

See Also

- Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.3 Wizard Scripts

A wizard script is a script created using the Script Wizard feature of Script Author. Like graphical scripts, wizard scripts are executed at runtime in any Scripting Engine interface.

Panels are the only script objects to display at runtime. Each panel in a wizard script contains at least one question user interface control (and may contain as many as required). Panels may also contain panel text.

Each panel in a wizard script includes a Continue button, which must 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 wizard 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 script developers determine flow by making choices in wizard windows for panels and answer choices. Based on these, Script Author associates default or distinct branching as appropriate into the script's metadata. Wizard scripts do not include conditional or indeterminate branch types.

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 objects and branch types accessible to the graphical script user interface.

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.

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

However, after creating a wizard script, you can graph the script and customize any panel to contain these elements if required.

To add any element to a wizard script that is not accessible from the Script Wizard (checkbox or button question controls; groups or blocks; global commands; panel layout elements such as graphics, hypertext links, or embedded values; or conditional or indeterminate branches), or to modify formatting for panel layouts (for example, to include differing horizontal alignment of question controls or text), you can convert a wizard script to a graphical script. This is also required if you want to change a default global property (disable footprinting, answer collection, or

the suspendable property of a script). Any modifications you make to the graphed script *cannot subsequently be viewed using the Script Wizard tool.* Thereafter, if you open the wizard script, the modifications will not appear.

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.

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 Script Author commands other than for panel validation. Thus, wizard scripts do not branch based on the evaluation of Script Author 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 Script Author 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.

See Also

- Scripts
- Graphical Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.4 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** 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.

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.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph

- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.5 Questions

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.

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

Answers, Answer Definitions, or Nodes Are Now Questions

Question user interface controls are the controls you select in Script Author when developing a script. The question UI control types available include (in the order presented in the UI for a graphical script) text field, text area, radio button, check box, button, drop-down list, password field, checkbox group, and multi-select list box.

If you select Button (in a graphical script only), you can include no other questions in the panel.

In the past, question UI controls have sometimes been referred to as *nodes*, *answers*, or *answer definitions*. For example, in the graphical Script Author UI, when defining a question UI control, you must select Answers from the panel properties tree. The window in which UI controls are designated is also still referred to as the Answer Entry Dialog window.

In keeping with newer terminology adopted with introduction of the Script Wizard feature, and in the interest of clarity, these are now referred to in product documentation either as questions or as question UI controls.

Wizard Question UI Limitations

Panels created with the Script Wizard do not include the single checkbox or the button (also referred to as the *submit button* or *push button*) question user interface controls.

Panels with Two or More Questions Include Generated Continue Button

When displayed at runtime, each panel with two or more explicitly defined questions also displays a Continue button. At runtime, script end users 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, question definition for panels *with no questions explicitly defined* is performed for you automatically. In other words, if you do not specify a question control, the application automatically creates a question, which displays a Continue button at runtime. If you graph the script and examine the panel, you see that the panel contains a question named Continue. This question has a label for reporting of Continue. In the data dictionary for this question, a specified lookup value set (for both the value passed to the application, and the display value) is also Continue.

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.

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.

Some Question Types Require a Response at Runtime

At minimum, for each panel, the end user must click the "Continue" button to progress to the next panel. Some questions, based on the selected UI type, may require additional end user input. For example:

- The button question control requires the end user to click prior to accepting the request and progressing to the next panel. This is *most often* used as a Continue button. In this case, the end user clicks "Continue" to progress to the next panel.

  In a graphical script, if multiple sets of values are provided in the data dictionary for the button question control, then an equivalent number of buttons will appear at runtime. Each answer choice appears in a horizontal row, which wraps to a second and subsequent line if required. (Generally, when defining multiple answer choices for a button UI control, you should severely restrict the amount of text per answer choice, for the most pleasing visual effect at runtime.) Only one of the buttons can be clicked at runtime to progress to the next panel.

- The radio button question control requires the end user to select one of the displayed radio buttons prior to accepting the "Continue" request and progressing to the next panel.

- The drop-down list control requires the end user to select one of the displayed answer choices from the list prior to accepting the "Continue" request and progressing to the next panel.

Question Properties

Each question may have several properties, as described in the table below. For each question defined in Script Author per panel, one question UI control appears in that panel at runtime.

This is true even if the script is created using the Script Wizard. These properties are evident for graphed wizard scripts.

Property	Data Type	Required?	Function or Description	Restrictions
Default for Distinct Branching	Check box	No	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.
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 question UI 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 question UI control in 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.
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.

Required Question Properties

- The properties required depend in part upon the selected question UI type. The only required property for all questions is Name. If you create a question without a name, Script Author will provide a unique name such as

untitledNode[n], where [n] is a sequentially provided integer unique to that script.

- *Every panel that uses distinct branching* requires one of the questions to be defined as the trigger for distinct branching. (This requirement was once true of each panel, regardless of whether distinct branching was used.) Distinct branching cannot be triggered on panels containing multi-select question controls, since more than one of the lookup values (answer choices) may be selected. Thus, this property is unavailable to the multiple-selection question UI types (multi-select list box and checkbox group).
- Panels in graphical scripts containing radio button, button, drop down, checkbox group, and multi-select list box question UI types all require answer choices (lookup values) to be defined by the script developer. Lookup value types include hard-coded specified values, as well as table, cursor, or command lookups. These are accessed in the data dictionary for a specific question by accessing **Answer Entry Dialog > Data Dictionary > Lookups**.

WYSIWYG Architecture Restrictions for Question Property Changes

1. Once the question properties listed earlier are saved, changes to question 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:

a. 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.

b. 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>".

c. 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 > **Jar Listings** subtab using the Scripting Administration console.

Question User Interface Types

Script Author provides nine question UI types, consisting of a familiar set of question UI 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.

UI Type	End User Action	Supports Null Value?	Supports Multi-Select Values?	Requires Answer Choices?	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
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

Each question UI type behaves according to its own specifications. Some characteristics are shared by similar question UI types. For example:

- Text field, text area, and password field question UI types all accept keyboard entry from the user. While the UI control differs in appearance or behavior for each, each type will accept up to 4,000 characters. This restriction is related to the character limit placed on the FREEFORM_STRING column of the IES_QUESTION_DATA table in the Oracle Applications database.
- Radio buttons and dropdown lists require an answer choice to be selected prior to clicking the default Continue button at runtime.
- Radio buttons, buttons, dropdown lists, checkbox groups and multi-select list boxes all require lookup values to be defined by the script developer.
- Radio buttons, check boxes (both individual and checkbox groups), buttons, and dropdown lists (single and multi-select types) all require actions by the user (a mouse click or a keyboard command) for answer choice values to be selected (assuming no default command is defined for the panel answer).
- Both multi-select question UI control types (checkbox groups and multi-select list boxes) can accept null (unselected) as a valid response.

- Both checkbox types (single and checkbox groups) can accept null (unselected).
- Of all question UI types, only the single checkbox 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).
- Of all question UI types, only the button type does not display any value entered into the Label for Reporting field. In reports generated, this value still appears.

Guidance on Using Question UI Types

Guidance on when to use specific question UI types follows below, along with UI type-specific information and recommendations.

UI Type	Description and Recommendations for Use
Text Field	Use when soliciting a short text response from the script end user at runtime. Without modifying panel HTML, the text field length typically displays between 14 and 22 characters, based on character set and font size. Longer answers can be input by the end user at runtime but the entire string is not visible. As text is added by the script end user beyond the visible set of characters, the scroll control enables and focus remains on the final portion of data entry.
Text Area	Use when soliciting several words or sentences of input from the script end user at runtime. Without modifying panel HTML, the text field area at runtime in the agent interface provides approximately 30 EM spaces of width and approximately four lines of depth. As more text is added by the script end user, the scroll control enables and focus remains on the line of data entry. A label may be added to this field but is not required. Labels are recommended if the user input desired is not explicitly clear (for example, specified by panel text), or if there are more than one text input controls into which to enter data in the panel.
Radio Button	Use for a series of conditions for which only one is allowed. The radio button control requires lookup values to be entered by the script developer in order to be functional at runtime. A single lookup value (option) must be selected by the user or an error condition Survey Administration Console. Specifically, message condition RequiredFieldUnanswered will indicate "Radiobutton group: <questionname> needs to be answered." A label may be added to this field but is not required, as the function of this control is generally evident.
Check Box	Use to evaluate a simple, single Boolean condition. Either this condition is **true**, in which case the end user selects this control at runtime, or it is **false**, in which case the control is left null. The check box control requires a value to be entered into the label for reporting. This question UI control differs from all others in this respect: all other click-based question UI types require one or more choices to be defined as lookup values (answer choices). This UI type will not display answer choices. Thus, label is an absolute requirement for this question UI type. This question UI control displays a checkbox at runtime that the end user may select or leave null. The selected checkbox passes a value of "true" and an unselected checkbox passes a value of "false."

UI Type	Description and Recommendations for Use
Button	Use to progress the script to a subsequent panel in the default flow when no specific choices are required of the end user. This requires a single lookup value to be defined. The display value for the button is typically "Continue." (Other lookup values can be used, at the script developer's discretion). Panels with the button question UI type do not contain the automatically generated "Continue" button that appears with all other UI types. A button question UI type can also contain multiple lookup values. Each lookup value displays as one of a horizontal row of buttons. At runtime, the script progresses when the end user selects one of the button choices. When used with a single lookup value, the button control specifically provides the script end user a method of moving to the next panel, which in Script Author uses default branching. When used with multiple lookup values, the button control provides the dual functions of providing an answer at runtime and progressing the script. The label property does not appear in panels displayed at runtime using the button UI type. However, this value is used in the IES schema, and is associated with the appropriate question in any reports generated. This question UI type can only be used in a panel with a single question. A panel with the button UI question type cannot contain any other question UI types.
Drop Down	Use for a series of conditions for which only one is allowed. The drop down control requires lookup values to be entered by the script developer in order to be functional at runtime. At runtime, the message "select one" appears by default. The lookup values available for end user response are not visible without end user action (mouse click or keyboard command to display values). A single lookup value (option) must be selected by the user or an error condition Survey Administration Console. Specifically, message condition RequiredFieldUnanswered will indicate "Drop Down "<questionname>": A valid choice must be selected." A label may be added to this field but is not required, as the function of this control is generally evident. Note that the drop down control is similar to the multi-select list box control, except this control does not support null values.
Password	Use for short amounts of text to be entered when you do not want the characters entered to display, generally for the password portion of user authentication. The password field at runtime spans the width of the user's work area for a single line. Any characters entered into this field display as an asterisk. Beyond the visible length, as more text is added by the script end user, focus continues to shift to the right. However, as the character displayed is always an asterisk, this is not immediately evident. A label may be added to this field but is not required. If more than one question in a panel uses this type, labels are strongly recommended for each. Labels are also recommended if the user input desired is not explicitly clear to the user. Recommend using validation in the question's data dictionary (General tab). This requires inclusion of a command, for example to verify the number or format of characters input, or to compare a user-supplied password against a validation table.

UI Type	Description and Recommendations for Use
Checkbox Group	Use for a series of conditions for which zero, one, or many selections are allowed. The checkbox group control requires lookup values to be entered by the script developer in order to be functional at runtime. This question UI control is similar to the single checkbox in appearance of the control (check boxes) and also that it supports null values (each check box may remain unselected and allow the end user to continue to the next panel). It differs from the checkbox in several ways. First, its values are determined from lookup values rather than the Label for Reporting value. For each lookup value defined, a separate check box appears at runtime for the end user to select if relevant. Second, the checkbox group supports multiple values. All checkboxes in a checkbox group are considered a single answer. The response passed to the Scripting Engine (and saved, if answer collection is enabled) is a vector. The vector can be null (if no selections are made by the end user). If any boxes are checked by the end user, then each corresponding value is included in the vector. Thirdly, the Label for Reporting functions as a label that appears to the left of the lookup values, similar to the label's functionality with the radio button and drop down question UI controls. Each lookup value appears to the right of its corresponding checkbox, in a horizontal row (which wraps if enough room is not available). Finally, this question UI control does not allow the "Default for distinct branching" property. Distinct branching cannot be triggered on multi-select questions, since more than one of the lookup values may be selected. Correspondingly, the Default for distinct branching control in the Answer Entry Dialog window will dim once this question UI type is selected, indicating its unavailability.
Multi-Select List Box	Use for a series of conditions for which zero, one, or many selections are allowed. The multi-select list box control requires lookup values to be entered by the script developer in order to be functional at runtime. The width of the multi-select list box control at runtime spans the width of the user's work area at runtime. Up to three lookup values are visible, with a vertical scroll control to display more if applicable. To select multiple values, script end users at runtime hold down the Control key and click once on each desired selection. To select multiple sequential values, the Shift key can be held. This question UI control does not allow the "Default for distinct branching" property. Distinct branching cannot be triggered on multi-select questions, since more than one of the lookup values may be selected. Correspondingly, the Default for distinct branching control in the Answer Entry Dialog window will dim once this question UI type is selected, indicating its unavailability. For this question UI type, the lookup values available for end user response are all visible without user action.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Graphical Script Objects
- Branches

- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.6 Graphical Script Objects

Scripts consist of configurable and nonconfigurable objects connected by branches. Objects are processing units that tell the script what to do. Branches are processing units that tell the script where to go. 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(groups and blocks) serve as containers for other graphs. These are referred to as subgraphs. These objects serve specific functions, but do not display at runtime.

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. 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, groups, and blocks.

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.

When displayed at runtime, each panel includes a Continue button. At runtime, script end users must click the Continue 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).

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 in a graphical script an imported script.

By default, wizard scripts contain a single group called Disconnect. 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.

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.

Some enterprises require the collection of 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 Corporation 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.

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

A block represents a child graph (a subgraph); as such, the graph represented by the block object must meet certain branching and termination requirements (see Minimum Requirements for Any Graph).

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.

Note: The single exception to the requirement for a termination node on every graph is a graph that contains an Indeterminate branch. See Indeterminate Branch Exceptions in this document.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.7 Branches

Branches are processing units that tell a script where to go. A branch is used to connect objects on the canvas in a graphical script. The type of branch you use determines the next object in the script flow. There are four types of branches:

- **Default:** The destination is the designated object.

  If there is more than one branch, 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 in a panel at runtime.

  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.

This branch type requires, as a prerequisite, one or more answer choices (or lookup values) to be defined in the data dictionary for a single question in the panel. For each distinct branch (with a specified value) drawn on the canvas, one distinct answer choice is designated, to indicate the correct flow at runtime when the specified response is provided by the script end user.

Values from a question's data dictionary cannot be specified in a distinct branch unless the question in the originating panel is first designated as the default for distinct branching (this is a Boolean option). Only one question per panel can be used as the default for distinct branching.

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

  This branch type requires the definition of a Boolean expression to be associated with the branch in Script Author and corresponding Java code to provide the expression to be evaluated at runtime. The branch is taken at runtime if the Boolean expression is determined to be true.

  Note: To avoid an error at runtime, any object using conditional branching should also include a default branch or indeterminate branch, to handle script flow if the Boolean expression at runtime is determined to be **false**. The conditional branch should have the appropriate branch order to ensure it is evaluated at runtime (and the Boolean expression is evaluated). The lower the branch order, the higher the priority of evaluation of the branch from the source object on the script at runtime.

- **Indeterminate:** The destination is based on the evaluation of an expression, the result of which 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.

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 will be evaluated. Following the procedures discussed in the topic Reordering Branches, this order can be changed 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.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.8 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 sub-graph 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.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java

- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.9 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. Shortcut panel

From the Script Author Java applet, you can view and update all modifiable global script properties of a graphical script from the File menu (**File** > **Script Properties**) or by right-clicking on an empty portion of the canvas (**Edit** > **Edit Blob Properties**). From the Script Wizard, you can view and update all modifiable global script properties in the Define Script Properties window.

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
Answer Collection	Boolean	N/A	TRUE	TRUE
Suspendable	Boolean	N/A	TRUE	TRUE
Pre-Actions	Null	N/A	Null	N/A
Post-Actions	Null	N/A	Null	N/A

Global Script Property	Graphical Script UI Type	Wizard Script UI Type	Graphical Script Default Value	Wizard Script Default Value
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. The script frame is a Java applet wrapped in an Oracle form. 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 runtime Java applet 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. You cannot open graphical scripts using the Script Wizard, nor can you open wizard scripts using Script Author graphical tools; however, you can *graph* a wizard script, creating a copy of the wizard script in graphical script form (with a script type of graphical script). The script name for the new, graphed script is identical to the wizard script, but is prepended with "Copy of." Scripts of both types can be listed (or removed from the database) from the Scripting Administration console.

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 meaning 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 (for example, regarding the purpose of the script, development intentions, or to identify authorship), and to allow those comments to be viewed by other script developers. 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. These fields are only visible from the Script Properties window of the graphical script or the Define Script Properties window of a wizard 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. Default selection is AMERICAN, the appropriate option for English spoken in the United States.

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 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 by clearing the default selection in the Collectable? checkbox (from the Answer Entry dialog for a specified answer, **Edit Data Dictionary > General Tab > Collectable?**).

Suspendable

The Suspendable check box option is visible only from a graphical script, in the script properties dialog (**File** > **Script Properties**). This option is selected by default.

Wizard scripts also default this option to "true." There is currently no method for changing the suspendable feature for a wizard script other than to convert the script to a graphical script.

When the Suspendable option is checked, and the IES : Display Suspend Button on Script Frame profile option is set to **true**, the Scripting Engine agent interface displays a Suspend button on the bottom of the script frame. This button is hidden if the profile is set to **false** (or remains null). When this button is clicked by an agent during script runtime, the current script interaction is suspended.

Suspended script interactions can be resumed from the Navigator by an Oracle Applications user with an appropriate responsibility to access Oracle Scripting scripts. 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.

Only scripts executed in the Scripting Engine agent interface currently support suspension of script interactions.

Pre-Actions and Post-Actions

Global script actions (pre-actions and post-actions) are Script Author 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 property causes a script information area to appear below the product identification (Oracle Scripting 11i). This takes objects of two data types: text and timer. Timers require Java commands to enable at runtime. Each static panel may contain up to nine data elements. Each can be provided with dynamic features by associating commands.

Shortcut Panel

Below the script information area (if relevant) and immediately above the panel display area, the shortcut button area displays shortcut buttons, which may be associated with any Script Author command. The shortcut button area was formerly known as the shortcut button bar or the shortcut panel.

See Also

- Scripts

- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.10 Oracle Scripting Users

This section includes the following topics:

- Script Author Users
- Scripting Administrative Users
- Scripting Engine Users
- Survey Administrative Users

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph

- Global Script Properties
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.10.1 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 Corporation strongly recommends that only trusted users be provided with the Scripting Administrator responsibility.

See Also

- Scripting Administrative Users
- Scripting Engine Users
- Survey Administrative Users

1.2.2.10.2 Scripting Administrative Users Users of the HTML-based Scripting Administration console include script developers (see Script Author Users), 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.

See Also

- Script Author Users

- Scripting Engine Users
- Survey Administrative Users

1.2.2.10.3 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, Oracle Collections, 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 11*i*-certified Web browser.

See Also

- Script Author Users
- Scripting Administrative Users
- Survey Administrative Users

1.2.2.10.4 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.

See Also

- Script Author Users
- Scripting Administrative Users
- Scripting Engine Users

1.2.2.11 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 meta-data 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).

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph

- Global Script Properties
- Oracle Scripting Users
- Using Custom Java
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.12 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.

Using the caching architecture of Oracle Scripting, custom Java code is executed by Oracle JInitiator on the client workstation (for the agent interface). 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.

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 Corporation 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.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Java Applet Versus Standalone Application
- Shortcuts

1.2.2.13 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.

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java

1.2.2.14 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.

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.

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).

See Also

- Scripts
- Graphical Scripts
- Wizard Scripts
- Differences Between Wizard and Graphical Scripts
- Questions
- Graphical Script Objects
- Branches
- Minimum Requirements for Any Graph
- Global Script Properties
- Oracle Scripting Users
- Custom Code
- Using Custom Java
- Java Applet Versus Standalone Application

1.2.3 Script Author Features

The component of Oracle Scripting that provides script developers with tools to create, modify, and deploy scripts is referred to as Script Author. Script Author is a

Java applet launched from the Scripting Administration console. Using this authoring environment, trained developers build scripts to be executed using the Scripting Engine. Scripts created using the graphical tools are referred to as graphical scripts. Scripts created using the Script Wizard feature of Script Author are referred to as wizard scripts. Custom code can be associated with graphical scripts globally, or at certain points in the script (corresponding with flow of the script at runtime). This code can execute a variety of customized or predefined commands to extend the functionality of the script.

This section includes the following topics:

- Script Author Graphical Layout
- Panels
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Script Wizard
- Data Dictionary
- Script Author File Management

See Also

- Script Author Terminology
- Script Author Concepts

1.2.3.1 Script Author Graphical Layout

The Script Author is a graphical layout environment. Script developers build graphical scripts by selecting *objects* from one of two tool palettes, and placing them on the work area (the canvas). Objects are connected by *branches*.

Objects include configurable objects (panels, groups and blocks) and nonconfigurable objects (start and termination nodes). Objects have properties which instruct the Scripting Engine how to behave when each object is reached in the flow of a script.

Branches direct the flow of the script at runtime (they tell the Scripting Engine where to go next), based on instructions received in the script. Flow in the progression of the script is controlled by branch type, and can also be designated

through the evaluation of actions programmed into the script. Actions contain Script Author commands. Branch types include default, distinct, conditional, and indeterminate branches. Each enforces different business rules to control flow of the script.

Project and Syntax Tab

The left side of the Script Author development environment displays either a Project tab or a Syntax tab.

The Project tab contains a table displaying properties of the currently selected object. Above this table, you can list objects on the canvas using one of four views:

- The script view displays all objects as defined on the canvas, including their hierarchy.
- The group view displays an alphabetical listing of groups defined in the script only.
- The block view displays an alphabetical listing of blocks defined in the script only.
- The panel view displays an alphabetical listing of panel defined in the script only.

The Syntax tab provides a list of each object on the canvas that breaks Script Author syntax rules.

Debug Pane

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.

Status Bar

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.

Hierarchical Graphs

Scripts consist of one or more graphs (collections of Script Author objects). Each graph automatically includes a *start node* which indicates where runtime flow begins. No properties can be associated with a start node (it is a nonconfigurable object). A start node is represented in a graphical script by an equilateral triangle.

Each graph contains a termination node, represented in a graphical script by a red octagon. As the shape suggests, it represents the end of processing. Typically, each graph contains at least one termination node and may contain any number (you can include separate termination nodes for any specific flow on a graph).

Processing of a script in any graph begins with the start node on the root graph, and proceeds to evaluate each configurable object (panels, groups, and blocks) in the chosen path for that script session. The script session at runtime ends when the termination node on the root graph is reached.

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*. The graph from which a sub-graph 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.

See Also

- Panels
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Script Wizard

- Data Dictionary
- Script Author File Management

1.2.3.2 Panels

A panel is the primary object of a script, and is the most configurable object. A panel is represented in a graphical script by a blue sphere. The panel contains the information that is displayed by the Scripting Engine at runtime (in the panel presentation area, for the Scripting Engine agent interface, or in the browser window, for the Scripting Engine Web interface). At runtime, each panel displays a button at the bottom of the panel. When the end user clicks the button, this registers the end user's responses to the questions in the panel, and progresses the script to the next panel.

Panels are created and edited using the Script Wizard or graphical tools. Regardless of which method was used to create it, the 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). 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.

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
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)

Graphical Script	Wizard Script	End User Action 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.

See Also

- Script Author Graphical Layout
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor

- Script Wizard
- Data Dictionary
- Script Author File Management

1.2.3.3 Groups

A group is a configurable object represented in a graphical script by a green cube clustered with three small colored spheres. A group represents a sub-graph in a graphical script. Unique properties of a group include *shortcuts* and *permissions*.

Groups are processing units only. Any commands associated with the group are executed when reached in the flow of the script. If a group contains panels, those panels display at runtime when reached in the flow of a script transaction. However, no aspect of a group is displayed at runtime.

Each group can contain a single shortcut, which is a name property. Shortcuts can be used at runtime to route the flow of a script to the group containing the specific shortcut. This requires use of the jumpToShortcut API.

Permissions are not implemented and should not be used. No support is available for this feature.

The primary purpose for a group is to provide organization for a script by grouping related functionality. It provides the ability to invoke its functionality (like a callable subroutine) from any point in the script using a shortcut.

See Also

- Script Author Graphical Layout
- Panels
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Script Wizard
- Data Dictionary
- Script Author File Management

1.2.3.4 Blocks

A block is a configurable object represented in a graphical script by a single green cube. A block represents a sub-graph in a graphical script. Unique properties of a block include its type and its object dictionary.

Blocks are processing units only. Any commands associated with the block are executed when reached in the flow of the script. If a block contains panels, those panels display at runtime when reached in the flow of a script transaction. However, no aspect of a block is displayed at runtime.

Each block can be designated by type to perform a database action (insert, query/select, or update), or to serve as a clearly designated container for one or more APIs.

For database functions, the object dictionary is used to determine connection information, join conditions, and query data such as constraints and columns.

See Also

- Script Author Graphical Layout
- Panels
- Groups
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Script Wizard
- Data Dictionary
- Script Author File Management

1.2.3.5 Graphical Script Object and Branch Properties

The two nonconfigurable objects in every graph of a graphical script are the start node and the termination node. These have no names or configurable properties associated with them.

Each configurable object and every branch created using Script Author graphical tools has properties associated with it. These include a name property and a comments property. The comments property is blank by default. In each graphical script, every configurable object and branch placed on a graph is automatically assigned a unique default name. Script developers should replace default names for objects with meaningful names. It is also recommended to provide meaningful

names to distinct, conditional, or indeterminate branches to provide an indication of each branch's function. This is not required for default branches; names assigned to default branches are not displayed in Script Author.

The script itself is considered a configurable object. Thus, it too has a name and comment property. These are referred to as global script properties. The global script name is the name under which the script is stored in the database when deployed from Script Author. This must be distinguished from the name assigned to the script file, which may differ.

When saving a graphical script to the filing system, the Script Author applet automatically provides a file extension of .SCRIPT if no file extension is provided by the user. Script Author can open files saved with both .SCR and .SCRIPT file extensions.

> **Note:** When opening a graphical script saved without a file type designation of .SCR or .SCRIPT, you must select **File Type > All Files (*.*)** from the Script Chooser (the open script window).

Object properties for every object and branch can be accessed in one of two ways:

1. By double-clicking on the object.
2. By right-clicking on the object and selecting **Edit Blob Properties** from the pop-up menu. This option is not available on the Macintosh platform.

Additionally, properties can be assigned to the global script by selecting **File > Script Properties**.

In order to be syntactically correct, all objects on a particular graph must have properties associated with them, and be connected with appropriate branching from the start node to the termination node. Panels must contain at least one *question*. Groups and blocks must contain proper termination of their child graphs.

See Also

- Script Author Graphical Layout
- Panels
- Groups
- Blocks
- Panel Layout Editor

- Script Wizard
- Data Dictionary
- Script Author File Management

1.2.3.6 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 re-imported 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 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. *Instructional text* indicates to the script end user specific instructions. *Spoken text* indicates text 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.

See Also

- Script Author Graphical Layout
- Panels
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Script Wizard
- Data Dictionary
- Script Author File Management

1.2.3.7 Script Wizard

As of Oracle Scripting release 11.5.9 or later, or Interaction Center Family Pack Q or later, Script Author includes a Script Wizard. 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. At this time, 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.

Using a SQL tool such as SQL*Plus, you can list properties for saved or deployed scripts. The metadata for saved scripts are stored in table IES_DEV_SCRIPTS, and the metadata for deployed scripts are stored in table IES_DEPLOYED_SCRIPTS. To differentiate between script types, wizard scripts are assigned a SCRIPT_CATEGORY value of 0, whereas graphical scripts are assigned a SCRIPT_CATEGORY value of 1.

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 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.

See Also

- Script Author Graphical Layout
- Panels
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Data Dictionary
- Script Author File Management

1.2.3.8 Data Dictionary

Each question in a panel has a data dictionary associated with it. For question UI types 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.

Data Dictionary and Wizard Scripts

The following are methods by which the data dictionary for a question is populated when you create a wizard script:

- Providing answer choices using the Answer Manager in the Script Wizard populates a question’s data dictionary.
- Changing any defaults in the Script Wizard Define Question Detail window causes a default command to be associated with the specified question's data dictionary.

- Specifying validation in the Script Wizard Define Question Detail window validation causes answer validation to be associated with that question in the panel.

Data Dictionary and Graphical Scripts

In graphical scripts, you can access the data dictionary for any question. This window has three tabs: General, Table info, and Lookups.

In the General tab, you can:

- Deselect the default Boolean Collectible property, ensuring that answer collection does not record the end user's response to this question.
- Associate answer validation for this question by defining a Script Author Java command.
- Associate a Script Author default command with this question.
- Identify a data source for this question.

In the Table info tab, you can identify table, column, format mast and data type information for a question.

In the Lookups tab, you can:

- Associate a specific table for a table lookup.
- Specify a cursor lookup, which will populate the answer choices with the current value of the scripting cursor at that point in the script at runtime.

  The cursor is populated by any SQL select statement, and is flushed each time a new query is executed.
- Associate a command lookup for this question, specifying a Script Author command.
- Specify hard-coded answer choices for this question.

Accessing the Data Dictionary in a Graphical Script

Access the data dictionary as follows:

1. In the Panel tree, select **Answers**.
2. In the Answers pane, click **Add**.

   The Answer Entry Dialog window appears.
3. Click **Edit Data Dictionary**.

The Edit Data Dictionary window appears.

See Also

- Script Author Graphical Layout
- Panels
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Script Wizard
- Script Author File Management

1.2.3.9 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 Corporation 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.

See Also

- Script Author Graphical Layout

- Panels
- Groups
- Blocks
- Graphical Script Object and Branch Properties
- Panel Layout Editor
- Script Wizard
- Data Dictionary

1.3 Understanding the Scripting Engine

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 Terminology
- Scripting Engine Concepts
- Scripting Engine Agent Interface
- Scripting Engine Web Interface

See Also

- Oracle Scripting Overview
- Understanding Script Author
- Understanding the Scripting Administration Console
- Understanding the Survey Administration Console

1.3.1 Scripting Engine Terminology

Following are definitions of some terms helpful in understanding the Scripting Engine. For more terminology, refer to the glossary.

Scripting Engine

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).

Scripting Engine agent interface

The Scripting Engine agent interface is an Oracle forms client that executes Script Author scripts at runtime. Users of the agent interface are interaction center agents; this interface is used in combination with Oracle business applications. The agent interface wraps several functional Java beans in an Oracle form to comprise the full agent user interface (UI). At the top of the UI, a toolbar appears, containing forward and back buttons. Panel contents display in the panel presentation area or panel display region; if programmed, information appears above this area at runtime in the script information area (formerly referred to as a "static panel"). Immediately below the script information area and above the panel display area, if programmed, buttons appear in a shortcut button area. To the right of the panel presentation area, a progress area shows the path through the current script session and (optionally) responses selected for each question per panel. These Java beans are all wrapped in a script frame that contains a progress indicator on the bottom left, a Disconnect button on the right, and (optionally) a Suspend button.

Scripting Engine Web interface

The Scripting Engine Web interface is the execution of a script in an Oracle Applications 11*i*-certified Web browser. Scripts executed using the Scripting Engine Web interface are typically used either as survey questionnaires, or as Web scripts launched from an integrated Oracle self-service Web application such as Oracle iSupport. Any script executed in the Web interface requires survey campaign administration. Unlike the agent interface, the only functional component displayed when executing a script is the panel presentation area. Whether programmed as part of the script or not, no script information area or shortcut buttons will appear in the Web browser. No progress area, toolbar, Disconnect button, or Suspend button appear anywhere in the Web interface. Previous panels can be revisited by clicking the Web browser's Back button. Forward progress is controlled by question user interface controls in each panel, including at minimum one submit-style Continue button.

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 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.

See Also

- Scripting Engine Concepts
- Scripting Engine Agent Interface
- Scripting Engine Web Interface

1.3.2 Scripting Engine Concepts

This section includes the following topics:

- Scripting Engine Function
- Scripting Engine Agent Interface
- Scripting Engine Web Interface
- Footprinting and Answer Collection
- What Displays in a Script at Runtime?
- Script End Users

See Also

- Scripting Engine Terminology
- Scripting Engine Agent Interface
- Scripting Engine Web Interface

1.3.2.1 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 application running in an Oracle form, and the Web interface, a sequential interpretation of a script with each panel represented by one JSP page rendered in an Oracle Applications 11*i*-compatible Web browser.

See Also

- Scripting Engine Agent Interface
- Scripting Engine Web Interface

- Footprinting and Answer Collection
- What Displays in a Script at Runtime?
- Script End Users

1.3.2.2 Scripting Engine Agent Interface

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

The Scripting Engine agent interface is designed particularly with the needs of interaction center agents in mind. Like the Scripting Engine Web interface, panels (that may contain text and images, and that *must* contain at least one question per panel) 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 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; and a programmable Disconnect button. Optionally, a Suspend button can also be displayed.

Agent User Interface Description

Several functional components are wrapped in an Oracle form to comprise the full agent user interface (UI). At the top of the UI, a toolbar appears, containing forward and back buttons. Panel contents display in the panel presentation area or panel display region; if programmed, information appears above this area at runtime in the script information area (formerly referred to as a "static panel"). Immediately below the script information area and above the panel display area, if programmed, buttons appear in a shortcut button area. To the right of the panel presentation area, a progress area shows the path through the current script session and (optionally) responses selected for each question per panel. These Java beans are all wrapped in a script frame that contains a progress indicator on the bottom left, a Disconnect button on the right, and (optionally) a Suspend button. The Scripting Engine agent interface executes as a Java servlet.

See Also

- Scripting Engine Function
- Scripting Engine Web Interface
- Footprinting and Answer Collection
- Script End Users

1.3.2.3 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. There is no progress area, script information area, button bar, or Disconnect button. End users can use the browser's Back and Forward controls to aid with navigation.

Executing a script in the Web interface first requires administration, in the form of establishing a set of survey campaign requirements using the Survey Administration console. These requirements include creating a survey campaign and defining an associated cycle and deployment, identifying JSP components known as survey resources, and identifying a deployed script as the questionnaire.

- 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 User Interface Description

The Scripting Engine Web interface is the execution of a script in an Oracle Applications 11*i*-certified Web browser. Scripts executed using the Scripting Engine Web interface are typically used either as survey questionnaires, or as Web scripts launched from an integrated Oracle self-service Web application such as Oracle iSupport. Any script executed in the Web interface requires survey campaign administration. Unlike the agent interface, the only functional component displayed when executing a script is the panel presentation area. Whether programmed as part of the script or not, no script information area or shortcut buttons will appear in the Web browser. No progress area, toolbar, Disconnect button, or Suspend button appear anywhere in the Web interface. Previous panels can be revisited by clicking the Web browser's Back button. Forward progress is controlled by question user interface controls in each panel, including at minimum one submit-style Continue button.

See Also

- Scripting Engine Function
- Scripting Engine Agent Interface
- Footprinting and Answer Collection
- What Displays in a Script at Runtime?
- Script End Users

1.3.2.4 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).

See Also

- Scripting Engine Function
- Scripting Engine Agent Interface
- Scripting Engine Web Interface

1.3.2.5 What Displays in a Script at Runtime?

Scripts are miniature programs created using Oracle Scripting's Script Author component, for execution at runtime using the Scripting Engine component.

Graphical scripts consist of configurable and nonconfigurable objects, connected by branches. Each object has its own associated properties.

Wizard scripts, although created by users answering a series of questions, adhere to the same principals of construction. If translated to a visible (graphical) script, a typical wizard script appears indistinguishable from a graphical script, with two notable exceptions. By default it excludes blocks, which are not currently part of the Script Wizard functionality. Graphed wizard scripts also include only one group (Disconnect, which contains the WrapUpShortcut shortcut property, enabling the Disconnect button in the agent interface). There are no panels in the automatically created Disconnect group.

For both Scripting Engine interfaces, of all Script Author graphical objects, only panels display at runtime. Other graphical elements (configurable objects such as groups and blocks; nonconfigurable objects, such as start and termination nodes; and branches) contain information that control processing of the script.

Each panel contains at least one question user interface control (at minimum, a Continue button). At runtime, when the script end user clicks the Continue button, the next panel in sequence is then displayed. Any container objects (groups or blocks appearing between two panels) are processed by the Scripting Engine at the corresponding time in the flow of the script. This process is seamless to the script end user. Wizard scripts will not reach any container objects in the flow of a script unless the Disconnect button is clicked in the agent interface.

Differences Between Scripting Engine Runtime Interfaces

Only panels display at runtime; each panel may contain predefined questions, answer choices, graphics and hyperlinks. In addition, each panel contains a Continue button.

For scripts executed in the Scripting Engine Web interface, each panel appears as a separate navigable page in the end user's Web browser. Clicking Continue at runtime leads the user to the next panel in sequence. Additionally, you can backtrack in the script using the browser's back button, and return to previously visited panels using the browser's forward button.

For scripts executed in the Scripting Engine agent interface, panel display is controlled by the system profile IES : Scripting Panel Display Mode. When set to single-panel display mode, like the Web interface, only the current panel appears in the window, and clicking Continue causes the next panel to display. When set for multiple-panel display, the active panel has focus, but previous panels are visible. Users can scroll through previously visited panels and click on a panel to make it active.

Although the panel display region in the agent interface is very similar in appearance and function to scripts executed in a Web browser, that is where similarities end. In the Web interface, the Web browser window frames the panel display region. In the agent interface, an Oracle form entitled "Oracle Scripting" serves as the scripting frame. This form serves as a wrapper containing several Java components. Chief among these is the panel display region described earlier. This provides the main functionality of a script at runtime, displaying panel contents and prompting the end user (an agent) for at least one response per panel. Included in the agent interface (and not the browser) is a column to the left of each panel enumerating the sequence in which panels are viewed in the current transaction. (Based on script requirements, this might be a unique path for each session.)

Other functionality unique to the agent interface includes a Progress area. This too lists the sequence of panels visited and (optionally) the question names and responses provided for each. Listing the responses provided to each question is the default behavior. For each script session, this can be toggled off by clicking the first button at the top of the Progress area, and on again by clicking the second button. Clicking on the panel name for a previous panel in the Progress area will pass focus to that panel. This provides perhaps the easiest method to backtrack in a script.

The remaining additional functionalities in the agent UI include a script information area (formerly referred to as a "static panel") and a shortcut button area (formerly "shortcut panel"). Both of these elements are blank unless explicitly programmed as a global script property. Without being defined, they appear above the panel presentation region as a blank line under the "Oracle Scripting 11i" label.

When defined, the script information area presents static or dynamic information in the form of text or one or more timers. Controlling data or timers requires Script Author commands defined in the Static Panel section of the global script properties window.

When one or more buttons is defined in the Shortcut Panel section of the global script properties window, a script at runtime will display each defined button. This appears as a horizontal row of buttons appearing immediately below the script information area and immediately above the panel presentation area. Each button enables a Script Author command to be associated with it. When enabled, a button

that is clicked at runtime performs the action (Script Author command) associated with it. Using Java methods referencing shortcut APIs, you can dynamically enable or disable buttons at different points in a script session.

The last part of the Scripting Engine agent interface that differs from the Web interface is the script frame itself, which contains:

- On the top left, a navigation toolbar (with back and forward buttons that follow the same rules as a browser, and a last visited panel button).
- On the bottom left, a status indicator.
- On the bottom right, a Disconnect button.

  For graphical scripts, this must be enabled by creating a group with the shortcut name WrapUpShortcut. For wizard scripts, this is automatically created.
- On the bottom right (immediately to the left of the Disconnect button), a Suspend button.

  This button appears when the IES : Display Suspend Button on Script Frame profile is set to True and when the Suspendable Boolean global property of a graphical script is true (selected, or enabled).

See Also

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

1.3.2.6 Script End Users

This section includes the following topics:

- Agent Users
- Self-Service Web Application Users
- Web Interface Guest Users

See Also

- Scripting Engine Function

- Scripting Engine Agent Interface
- Scripting Engine Web Interface
- Footprinting and Answer Collection
- What Displays in a Script at Runtime?

1.3.2.6.1 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.

See Also

- Self-Service Web Application Users
- Web Interface Guest Users

1.3.2.6.2 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.

See Also

- Agent Users
- Web Interface Guest Users

1.3.2.6.3 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.

See Also

- Agent Users
- Self-Service Web Application Users

1.3.3 Scripting Engine Agent Interface

The Scripting Engine agent interface is a Java-based user interface executed from an Oracle Applications session. Since this executes as a Java bean wrapped in an Oracle Form, this interface is variously referred to as the agent interface or agent UI, the Forms client, or the agent Java application.

Using the Scripting Engine agent interface, interaction center agents are guided through their interactions with customers. The business logic built into the script, including custom Java, PL/SQL, and other commands, guides the agent through the script. As the trained agent moves effortlessly through guided messages represented by script panels, the Oracle Scripting objects embedded in the script control complicated processing. The business logic embedded in the script (such as rules-based branching, data integration, and commands associated with specific objects and events) is evaluated dynamically, along with the responses provided by the agent, progressing or flowing through the script in a logical manner and accomplishing the exchange of information required by the transaction. Oracle Scripting scripts can currently be launched by interaction center agents from within three business applications in the CRM Suite: Oracle TeleService, Oracle TeleSales, and Oracle Collections. If the agent has the appropriate responsibility to access the business application, no additional (Scripting-specific) responsibilities are required.

Scripts can also be executed in this interface in "standalone" mode, without use of a calling business application. This requires the agent to log into Oracle Forms-based applications as a user with the Scripting User or Scripting Agent responsibility.

The Scripting Engine agent interface is simple to use and promotes rapid agent training. When a script is requested by an agent, the Scripting Engine launches as a Java bean in a separate Oracle Forms window. The window is described in the following sections.

This section includes the following topics:

- The Panel Display Area
- The Progress Area
- The Script Information Area
- The Shortcut Button Area

- The Disconnect Button
- The Suspend Button
- The Toolbar

See Also

- Scripting Engine Terminology
- Scripting Engine Concepts
- Scripting Engine Web Interface

1.3.3.1 The 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. The single exception is the case of a button control. A button question UI control, *may* contain a value other than "Continue." Alternatively, 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 next panel.

Note: If the script developer uses the "button" question UI control type in a panel, that panel may contain only a single question (or "node"). The button control is not available as a choice of UI control if another UI control type is used. After you select the button control, you are not permitted to add other questions to the panel.

See Also

- The Progress Area
- The Script Information Area
- The Shortcut Button Area
- The Disconnect Button
- The Suspend Button
- The Toolbar

1.3.3.2 The 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.

See Also

- The Panel Display Area
- The Script Information Area
- The Shortcut Button Area
- The Disconnect Button
- The Suspend Button
- The Toolbar

1.3.3.3 The 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*.

See Also

- The Panel Display Area
- The Progress Area
- The Shortcut Button Area
- The Disconnect Button
- The Suspend Button

- The Toolbar

1.3.3.4 The 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.

See Also

- The Panel Display Area
- The Progress Area

- The Script Information Area
- The Disconnect Button
- The Suspend Button
- The Toolbar

1.3.3.5 The 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).

See Also

- The Panel Display Area
- The Progress Area
- The Script Information Area
- The Shortcut Button Area
- The Suspend Button

- The Toolbar

1.3.3.6 The Suspend Button

For some scripts, a **Suspend** button appears 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).

See Also

- The Panel Display Area
- The Progress Area
- The Script Information Area

- The Shortcut Button Area
- The Disconnect Button
- The Toolbar

1.3.3.7 The 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.

See Also

- The Panel Display Area
- The Progress Area
- The Script Information Area
- The Shortcut Button Area
- The Disconnect Button
- The Suspend Button

1.3.4 Scripting Engine Web Interface

The Scripting Engine Web interface is essentially the runtime interface of a script in an Oracle Applications 11*i*-certified Web browser. The browser interprets a script in HTML as a series of Java server pages.

As the survey respondent or Web script user responds to the question UI controls on each page, she moves effortlessly through guided messages (represented by script panels). The Oracle Scripting objects embedded in the script control complicated processing. The business logic embedded in the script (such as rules-based branching, data integration, and commands associated with specific objects and events) is evaluated dynamically, along with the responses provided by the user, progressing or flowing through the script in a logical manner and accomplishing the exchange of information for which the script was designed.

To execute a script in a Web browser, users access a survey URL. To obtain a valid survey URL, a survey administrator must first set up a survey campaign, and activate a deployment belonging to the survey campaign. These steps are documented in *Oracle Scripting Implementation Guide*.

This section includes the following topics:

References

For more information, including detailed survey concepts, see the Detailed Product Description section in *Oracle Scripting Implementation Guide*.

See Also

- Scripting Engine Terminology
- Scripting Engine Concepts
- Scripting Engine Agent Interface

1.3.4.1 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.

See Also

- User Interface Components Differ
- Ramifications of User Interface Differences
- Survey Resources

1.3.4.2 User Interface Components Differ

The Web interface does not have all the UI elements contained in the Agent interface. For the survey respondent, the script content in each HTML page equates to the panel display area for the agent interface. The Web interface does not contain a progress area, script information area, shortcut button area, suspend button, or Disconnect button. Instead of the Oracle Applications toolbar, the browser's default toolbar appears.

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

See Also

- Uses for the Scripting Engine Web Interface
- Ramifications of User Interface Differences
- Survey Resources

1.3.4.3 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.

The Web interface does not allow suspension or resumption of script transactions.

See Also

- Uses for the Scripting Engine Web Interface
- User Interface Components Differ
- Survey Resources

1.3.4.4 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 OAF 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 OAF 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.

See Also

- Uses for the Scripting Engine Web Interface
- User Interface Components Differ
- Ramifications of User Interface Differences

1.4 Understanding the Scripting Administration Console

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

For Oracle Scripting release 11.5.10 (or Interaction Center Family Pack R), this component relies upon the Oracle CRM Technology Foundation (JTT) technology stack.

This section includes the following topics:

- Scripting Administration Console Features
- Oracle Scripting Administration Concepts

See Also

- Oracle Scripting Overview
- Understanding Script Author
- Understanding the Scripting Engine
- Understanding the Survey Administration Console

1.4.1 Scripting Administration Console Features

The Scripting Administration console is an HTML administration user interface for script developers and administrators. As of this release, this component 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
- Oracle Scripting File Administration
- Oracle Scripting Agent Interface Reports

See Also

- Oracle Scripting Administration Concepts

1.4.1.1 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.

See Also

- Oracle Scripting File Administration

- Oracle Scripting Agent Interface Reports

1.4.1.2 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.

See Also

- Script Author Applet
- Oracle Scripting Agent Interface Reports

1.4.1.3 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.

See Also

- Script Author Applet

- Oracle Scripting File Administration

1.4.2 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
- Scripting Administration Console View List
- Agent Interface Reports

See Also

- Scripting Administration Console Features

1.4.2.1 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.

See Also

- Scripting Administration Console View List

- Agent Interface Reports

1.4.2.2 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. 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.

See Also

- Scripting Administration Console
- Agent Interface Reports

1.4.2.3 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:

- Reports and Data
- Analysis and Tuning with the Panel Footprint Summary Report
- Required Report Parameters

See Also

- Scripting Administration Console
- Scripting Administration Console View List

1.4.2.3.1 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 Corporation 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.

See Also

- Analysis and Tuning with the Panel Footprint Summary Report
- Required Report Parameters

1.4.2.3.2 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.

See Also

- Reports and Data
- Required Report Parameters

1.4.2.3.3 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

See Also

- Reports and Data
- Analysis and Tuning with the Panel Footprint Summary Report

1.5 Understanding the Survey Administration Console

The Survey Administration console is an HTML administration user interface for survey and Web script administrators. From this administration console, you can create and administer survey campaigns and survey resources. Additionally, you can view individual responses received from an activated survey deployment using the Responses view of the Survey Campaign tab.

For targeted survey campaigns, you can also access Oracle Marketing functionality to administer marketing lists, and Oracle One-to-One Fulfillment functionality to create and administer fulfillment templates, including invitation and reminder master documents and their associated queries. These capabilities are all accessible from a single user interface for the convenience of survey administrators. While list management and fulfillment template management is hosted in the survey administration user interface, the functionality is provided and supported by the respective product teams.

This section includes the following topics:

- Survey Administration Console Features
- Survey Administration Concepts

References

- For administrative procedures regarding setting up survey campaigns, cycles, and deployments, or to view responses received, or for survey resource administration, refer to *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.

See Also

- Oracle Scripting Overview
- Understanding Script Author
- Understanding the Scripting Engine
- Understanding the Scripting Administration Console

1.5.1 Survey Administration Console Features

The survey component of Scripting was previously known as iSurvey.

The Survey Administration console is an HTML administration user interface for survey campaign administrators and managers. This console integrates functionality from various Oracle applications for the benefit of administering and monitoring survey campaigns. Oracle Scripting, Oracle One-to-One Fulfillment, and Oracle Marketing functionality are all accessible from this single user interface. The Survey Administration console has the following primary functions:

1. To administer survey campaigns, cycles, and deployments in support of a survey campaign. This is performed from the Survey Campaigns tab of the Survey Administration console. Additionally, you can view responses received from existing survey campaigns by viewing deployment detail in the response view.
2. To administer survey resources, which are JSP files that appear as header or footer sections, error pages, or final pages for scripts executed in a Web browser. This is performed from the Survey Resources tab of the Survey Administration console.
3. To perform list management supporting targeted survey deployments. This accesses Oracle Marketing functionality. This is performed from the Audience tab of the Survey Administration console.
4. To 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 for targeted survey deployments. This is performed from the Invitations tab of the Survey Administration console.

As of Oracle Scripting release 11.5.9 (or Interaction Center Family Pack Q), this component relies upon the Oracle Applications Framework (OA Framework or OAF) technology stack.

This section includes the following topics:

References

- For specific task-based steps to use the Survey Administration console, see the Survey Resources Administration Tasks and Survey Campaign Administration Tasks sections of *Oracle Scripting Implementation Guide*.
- Some task-based administrative steps for using the Audience and Invitations tabs of the Survey Administration console are als0 included in *Oracle Scripting Implementation Guide*, as they pertain to Oracle Scripting's survey functionality. Every effort is made to synchronize between the releases of Oracle Scripting and the various applications with which it integrates. Nonetheless, if the steps listed in *Oracle Scripting Implementation Guide* do not match the user interface you see when you attempt to perform a task using Oracle Marketing or Oracle One-to-One Fulfillment, please consult the product documentation appropriate to the release installed at your site.

See Also

- Survey Administration Concepts

1.5.1.1 Survey Campaigns Supported in Two Technology Stacks

When a survey campaign is established using the new Base Tech Stack option set to **OA Framework** (the new technology stack), the script (when executed in the Scripting Engine Web interface as a survey or Web script) runs using the new Oracle Applications Framework (OA Framework or OAF) technology stack.

If the Base Tech Stack option is set to **Deprecated - JTT**, the script at runtime executes in a Web browser using the Oracle CRM Technology Foundation (JTT) technology stack.

Oracle Corporation strongly recommends that all customers using the Scripting Engine Web interface plan and implement new survey deployments using the new Oracle Applications Framework functionality. The older technology stack is deprecated and will eventually be obsolete. Survey administrators select which technology stack new survey campaigns use at runtime by selecting the appropriate option from the Base Tech Stack list on the Create Survey Campaign page.

New features and functionality are available using the new technology stack, and will continue to be introduced. Most of these improvements will not be available to the older technology stack.

See Also

- Survey Campaigns, Cycles, and Deployments
- Survey Resource Administration
- The Survey Questionnaire
- The Survey URL

1.5.1.2 Survey Campaigns, Cycles, and Deployments

A *campaign* is a focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose.

Oracle Marketing uses a concept of a campaign that is required by business applications such as Oracle TeleSales, Oracle Collections, Oracle Advanced Inbound Telephony and Advanced Outbound Telephony. These are also known as marketing campaigns.

Oracle Scripting employs a similar concept to support its survey functionality, called the *survey campaign*. In the broad sense, a survey campaign is a focused effort by an enterprise to conduct an information gathering campaign by polling a defined market segment or population (customers accessing a Web site, members of a defined list, etc.) for a specific period of time (a cycle) for a particular business purpose. The purpose is to gather the information in the survey questionnaire for subsequent analysis and action. Typical actions might include the offering of a new product or service, or a change in business processes to improve satisfaction.

Survey campaign goals are achieved by two processes: creating a survey questionnaire (a script built with the campaign goals in mind), and in administering the survey campaign from the enterprise.

Since the primary purpose of the Survey component is to conduct a survey campaign, it uses data structures with names that reflect its purposes. 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. The survey campaign has two lower level data structures or child objects: cycles and deployments.

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.

A *deployment* contains specific details regarding execution dates for the survey campaign, and a deployment type. *Standard deployments* require no more information, and are executed by publishing, announcing, or embedding the resulting survey URL in e-mail messages or links. *Targeted 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).

See Also

- Survey Campaigns Supported in Two Technology Stacks
- Survey Resource Administration
- The Survey Questionnaire
- The Survey URL

1.5.1.3 Survey Resource Administration

At runtime, each panel in a script executes as a separate JSP page. If assigned to the 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; and upon completion, a final page resource displays, or the user is redirected to a URL designated as the final page.

Oracle Scripting supports two technology stacks for executing scripts in a Web browser. The base technology stack you select when creating a survey campaign determines the technology used to execute the script in a Web browser at runtime.

Oracle Corporation recommends using the OA Framework technology stack when possible.

Survey Resources and OAF

When using the OA Framework technology stack, all survey resources are HTML (not JSP). This technology stack removes the requirement (still in force for survey campaigns established in the JTT technology stack) to define survey resources. If you administer a survey campaign and elect not to specify resources, a default error page and final page is provided by the application at runtime, when appropriate.

If no header or footer is designated, each panel rendered in the Web browser will simply not include a header or footer section at runtime.

This technology stack also supports using a URL to redirect 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. (These can be HTML or JSP pages.)

This technology stack does *not* support the use of those survey resources defined in the survey administration console listed as type Deprecated - JSP.

Survey Resources and JTT

When using the deprecated JTT technology stack, all survey resources are JSP. HTML or redirect resources are not supported. At runtime, the script executes in a Web browser using the Oracle CRM Technology Foundation (JTT) technology stack.

- Each survey campaign set up in this technology stack still requires JSP survey resources to be designated as the header section, error page and final page.
- The footer section is optional.

See Also

- Survey Campaigns Supported in Two Technology Stacks
- Survey Campaigns, Cycles, and Deployments
- The Survey Questionnaire
- The Survey URL

1.5.1.4 The Survey Questionnaire

The survey questionnaire is a Script Author script created to exchange information between an enterprise and other parties. When used as a set of questions to gather feedback, obtain market data, tabulate opinion, measure satisfaction, or otherwise collect willingly provided survey data, it is a true survey questionnaire. If used to exchange information in other ways, it can also be called a Web script.

At runtime, the script designated at the survey campaign level as the survey questionnaire is executed in an Oracle Applications-compatible Web browser, using the Scripting Engine Web interface. This is often described as "running a survey." End users can include Oracle self-service Web application users executing a script in a browser from the application; Internet-enabled customers or prospects who have been provided or encounter a valid survey URL or a link to a valid survey URL; and members of an Oracle Marketing list who have received by e-mail an invitation or reminder to execute a survey in their browser. As the individual participating in a survey or Web script (sometimes referred to as the "survey respondent") moves effortlessly through each HTML page (corresponding to a single script panel in Script Author), the objects embedded in the script control complicated processing. The business logic embedded in the survey questionnaire script (such as rules-based branching, data integration, and commands associated with specific objects and events) is evaluated dynamically, along with the respondent's answers.

Fixed or Dynamic Flow

The respondent is guided through the panels of the script either through a rigidly prescribed order, or through a dynamically determined path in the survey. This is determined by the construction of the script, based on the needs of the enterprise or specific survey campaign requirements. Some survey campaigns or enterprises require the same information to be presented to or solicited from every customer in every transaction. Other survey campaigns or enterprises incorporate flexibility into the script, taking advantage of rules-based branching provided by multiple branch types, and using Scripting Blackboard APIs to enable selection and dynamic presentation of appropriate questions based on answers previously provided by the respondent in the survey or by information pulled into the script from database tables. This greatly enhances flow of the script at runtime, and yields better data and higher response rates.

In the background, the script executes as a servlet on the Apache Web server associated with an enterprise applications server or specified by survey administrators.

The script must be completed, tested, and deployed to the applications database to make it available as the survey questionnaire for a specific survey campaign in the Survey Administration console.

Who Creates a Survey Questionnaire Script?

Based on existing requirements for a specific business purpose, individuals trained in the use of Script Author develop the script, using graphical tools or the Script Wizard.

Generic surveys are often a series of simple questions with preset answer choices. Such surveys can in theory be created by relatively non-technical users who have been trained to use Script Author. For example, a script meeting this description can easily be created using the Script Wizard. However, Oracle Scripting provides very sophisticated functions for scripts, regardless of which Scripting Engine interface is used to execute them. If you want to use Script Author commands (including PL/SQL, Java, or blackboard commands), change global variables within the script, or customize panel HTML layouts, you must perform some script development using the graphical tools. Script developers are typically moderately technical functional users with access to other development resources with substantial technical ability and training in the required supporting technologies.

See Also

- Survey Campaigns Supported in Two Technology Stacks
- Survey Campaigns, Cycles, and Deployments
- Survey Resource Administration
- The Survey Questionnaire
- The Survey URL

1.5.1.5 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.

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 11*i*-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.
    - If this parameter uses the OA.JSP template, the deployment is created as part of a survey campaign with the Oracle Applications Framework

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 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 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 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 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.

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 three JSP templates, as described in the table below.

JSP Template	Tech Stack	Authentication	Description
OA.JSP	OAF	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	OAF	For targeted deployments, guest user only.	▪ Does not support menu-based hosting for *targeted* deployments. ▪ Uses applications guest user only.
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.

JSP Template	Tech Stack	Authentication	Description
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

- 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 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*.

1.5.2 Survey Administration Concepts

The survey component of Oracle Scripting allows enterprises to use Script Author scripts as Web-based survey questionnaires that can be executed in a Web browser (over the Internet or on an intranet). 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. Using the survey component of Oracle Scripting, enterprises can create, manage, and report on surveys to evaluate customer satisfaction, gain customer input on new initiatives, and gain other feedback from

survey respondents. The return data can then be used to improve product lines, target new or improved services, or otherwise improve responsiveness.

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 includes the following topics:

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

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.

See Also

- Survey Administration Console Features

1.5.2.1 Survey Administration Console

In order to execute a Script Author script in a Web browser using the Scripting Engine Web interface, survey administrators must define survey campaign requirements in an HTML administration console. This is true regardless of whether the script will be executed as a standard or targeted survey, or from a self-service Web application.

The Survey Administration console is an HTML user interface accessed from the Oracle Applications Personal Homepage (PHP) login by a user with the Survey Administrator responsibility. In this interface, a survey campaign is created and its dependencies (survey resources, Script Author script designated as the questionnaire, a survey campaign, cycle, and deployment) are defined.

Additionally, survey administrators can view individual results from a survey (ongoing or completed) from the response view of the Deployment Details page. Answers provided by respondents are stored in the Oracle Applications database schema.

Essentially, the Survey Administration console provides survey administrators the means to set up, execute, and monitor survey campaigns.

See Also

- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.2 The Survey Questionnaire

The survey questionnaire is a Script Author script created to obtain specific information. End users of the Survey component are customers or prospects viewing a survey questionnaire using any Oracle Applications 11*i*-certified Web browser, or individuals executing a Web script from an Oracle self-service application such as Oracle iSupport. As the individual participating in a survey ("survey respondent") or using the Web script moves through each HTML page (corresponding to a single script panel in Script Author), the objects embedded in the script control complicated processing.

On the end user's desktop, the user is guided through the script either through a rigidly prescribed order, or through a dynamically determined path (as determined by the custom script built based on the needs of the enterprise). The survey questionnaire script displays as a series of pages (one page per panel) in an HTML-based interface (the Web browser). The underlying technology differs slightly, based on the base technology stack used for execution. Survey campaigns using the OAF base 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.

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.

See Also

- Survey Administration Console
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources

- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.3 Survey Administration Console View List

When you view survey campaigns or survey resources in the Survey Administration console, the set of records which displays is filtered by the value in the View list. By default, only items created by the logged in user displays. To view items created by all users, change the value in the View list accordingly and click **Go**.

For example, when viewing survey resources, the default selection in the View list is "My Survey Resources." Correspondingly, the records that display on the page (if any) consist only of survey resources defined in the Survey Administration console by the Oracle Applications user account with which you are currently logged in. To display all survey resources (including those created by other users in this environment), select **All Survey Resources** from the View list and click **Go**. The Survey Resources page refreshes. The summary view table now lists survey resources (if any) created by all users for this environment. This is applicable to survey campaigns as well.

Note that if no objects of that category have been created, then the list headings will appear, with no entries. As soon as you create an object, it will appear in the refreshed list.

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL

- Concurrent Programs Supporting Survey Operations

1.5.2.4 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.

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.5 Survey Hierarchy, Levels and Objects

The Survey component of Oracle Scripting includes three hierarchical levels: survey campaigns, cycles, and deployments. Each level is a set of requirements to ultimately execute survey campaigns.

The collection of requirements for each level is stored as an object in the database. For this reason, survey campaigns, cycles and deployments are described as objects. Each existing survey object can be listed in the Survey Administration console.

When creating a survey campaign in the Create Survey Campaigns page, you also define a child object, the first cycle. You can subsequently modify the cycle name or create additional cycles for an open or active survey campaign from the Survey Campaign Details page.

Deployments are children objects to cycles. Deployments are created from the Create Survey Deployment page, and subsequently viewed and modified from the Deployment Details page.

You can also view responses to individual surveys taken for a deployment from the Deployment Details page by selecting the Response View option.

This section includes the following topics:

- Survey Object Dependencies
- The Parent-Child Survey Objects Relationship

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.5.1 Survey Object Dependencies General Survey object dependencies and rules include the following:

- Each survey campaign must have at least one cycle. It may have as many as required.
- Each cycle must have at least one deployment. It may have as many as required.
- Child objects cannot be shared, reused or inherited.
- 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.

- Before being executed, each deployment must be deployed (set to active status).
- The collection of requirements that is actually executed is the deployment, the lowest and most granular collection of execution parameters. Thus:
  - Deployments can be grouped into cycles to execute the same questionnaire and parameters over a different timeline for comparison purposes.
  - Cancelling a deployment does not affect the parent objects.

1.5.2.5.2 Parent-Child Survey Objects Relationship In order to execute a survey campaign, you must have at minimum three survey objects, directly associated in a parent-child relationship.

- You must define one survey campaign.
- You must create one cycle that belongs to (or is a "child" of) the survey campaign.
- You must create one deployment that belongs to (or is a "child" of) the cycle.

Each parent object must have at least one child in order to create executable survey campaigns. Each child object can have a "one-to-many" relationship with its parent object. Thus, a survey campaign must have one cycle and may have more than one, and each cycle must have one deployment and may have more than one. There is no limit to how many cycles or deployments can be created.

Why Allow One-to-Many Relationships?

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 of these principles, 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, and once afterwards. ABC customers are identified by three lists: direct mail customers, Internet customers, and retail customers.

Since the same survey will be executed in two separate time periods, company ABC survey administrators can choose to define two separate cycles (cycle 1 and cycle 2)

as child objects of survey campaign ABC. 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 (for example, January through March), and each deployment for cycle 2 also has identical start and end dates, for the second cycle of time (for example, 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 (for example, 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, for example, July through September.

No Sharing of Survey Objects

You cannot share or reuse existing survey campaign, cycle, or deployment objects. Nor can you copy an existing object and make modifications to one or more parameters. Even if you require a survey object (for example, a deployment) with the same set of parameters, you must redefine the object, associating it with its parent object (for cycles and deployments).

The only objects that can be shared and used are survey resources.

Therefore, in the preceding example, Cycle 1 and Cycle 2 will need to be created separately. Also, each of the three deployments for the three ABC lists will need to be created each twice, once for each cycle.

1.5.2.6 Survey Campaigns

The Survey component relies on the concept of a *campaign*: a focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose. The goal of a *survey campaign* is typically for an enterprise to obtain specific data by polling a target market segment or population for subsequent analysis and subsequent action. 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 campaigns have requirements that must be defined based on the campaign goals (for example: target population, purpose for obtaining the information, which information will be gathered, for what purpose, how long the campaign will be conducted, and in how many separate deployments of the same requirements).

The campaign goals are achieved by two processes: creating a survey questionnaire built with the campaign goals in mind, and in administering the campaign from the enterprise. The Survey Administration console supports these functions.

The survey campaign is the top-level object for creating and executing survey questionnaire campaigns. Survey campaigns are created using the Survey Administration console, the primary user interface for the Survey component of Oracle Scripting.

This section includes the following topics:

- Survey Campaign Dependencies
- Survey Campaign Status

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

Survey Campaign Dependencies

Each survey campaign requires:

- A script

  A script serving as the survey questionnaire must already be created and deployed to the applications database using the Script Author component of Oracle Scripting.

- A name to identify the survey campaign

- If executing surveys campaigns using the deprecated JTF technology stack, three JSP survey resources (header section, final page, and error page survey resources), as described in the Administering Survey Resources section of *Oracle Scripting Implementation Guide.*

  An optional footer section survey resource can also be defined. Each is a JSP page that must be defined in the Survey Administration console, and physically loaded to the $OA_HTML directory on the applications server.

  If executing surveys campaigns using the new Oracle Applications Framework technology stack, you can use HTML survey resources (for all resource types) or URL survey resources (for error page and final page resources). These can be defined and uploaded from the Survey Resources tab.

- At least one cycle

Survey Campaign Status

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 changes from open to active status. Once active, a survey campaign status may change to idle only. Its status never returns to open, nor can it be closed prior to changing to idle status. Active is only valid from the open or idle status. A survey campaign 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 was previously active and the deployment 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 cannot be manually set to idle by the survey administrator; this status is established by the system based on these business rules.
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 can be set to a status of 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 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 can be set to a status of 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.

Once created, and prior to defining a deployment, the status of a 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 is active or pending. From active, a survey campaign can become idle, and from idle to cancelled or closed. Cancelled and closed statuses only result from manual intervention by the survey

administrator. An active survey campaign can automatically change to idle based on events related to its deployments. For more information, see the section on deployment status.

Once all the deployments for a survey campaign have executed successfully, and the survey campaign is in idle status, you can close the survey campaign from the Status list by selecting **Closed** from the list. This indicates that the survey campaign has served its business purpose. 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, survey reports can be executed for analysis of survey returned data.

1.5.2.7 Cycles

A cycle is a child object of the survey campaign. It is the smallest set of requirements for survey campaigns, including only a cycle name, and is defined at the same time its parent object (the survey campaign) is created. Each survey campaign must have at least one cycle defined, and may have many.

The purpose for the cycle object is to provide the capability of grouping its children objects (deployments) for reporting purposes. When analyzing data from executed survey deployments, an enterprise can compare the results of one cycle (one grouping of deployments) to the results of another cycle executed at a different time. This comparative analysis provides the ability to quantify over time effectiveness, satisfaction, or other metrics related to the purpose for a survey campaign.

Cycles are defined from the Survey Campaign 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.

Cycle Dependencies

Each cycle requires a cycle name. Oracle Corporation recommends that each cycle be uniquely named within a single survey campaign.

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List

- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Deployments
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.8 Deployments

A deployment is the most detailed set of execution parameters. There are two types of deployments: targeted (previously referred to as list-based) and standard (previously referred to as non-list-based).

Targeted deployments use Oracle Marketing lists to send e-mail invitations (through Oracle One-to-One Fulfillment) to each member of the list, inviting them to participate in answering a survey questionnaire. These list members can be tracked by individual respondent.

Standard deployments are anonymous and have fewer requirements, as detailed below.

This section includes the following topics:

- Deployment Dependencies
- Targeted Deployment Dependencies
- Deployment Status

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns

- Cycles
- Survey Resources
- Prototypes
- Survey URL
- Concurrent Programs Supporting Survey Operations

Deployment Dependencies

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 Dependencies

For list-based deployments only, the following are also required:

- 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, then the following are also required:

- Reminder template name (reminder master document)
- Reminder e-mail message subject heading
- Number of reminders

- Reminder interval (in days)

Deployment Status

Status	Description
Open	A deployment 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	A standard deployment is active immediately after it is activated by the survey administrator. 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.
Cancelled	If a deployment has a status of open, its status 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 a deployment has an active status but there is a no interest on the part of the surveying enterprise to view the results, it can be set to incomplete status.
Closed	If a deployment is in active status and has run for its intended duration or has resulted in sufficient responses, it can be set to a status of closed, indicating that its purpose was fulfilled. Deployments in a pending or error status can also be set 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 will change to pending until they equal SYSDATE. 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 will change to active. For targeted deployments, the deployment details will 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.

Additional Requirements 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 will remain active if other deployments have been activated and have not encountered an error.

With survey campaigns for which the last deployment has been changed to 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 changes to idle, and can be 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.

1.5.2.9 Survey Resources

This section includes the following topics:

- Survey Resources Differentiated by Technology Stack
- Detailed Description of Survey Resources
- Creating Survey Resources
- Seeded JSP Survey Resources

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Prototypes

- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.9.1 Survey Resources Differentiated by Technology Stack Within the 11.5.10 release, the survey administration console supports two technology stacks for surveys executed at runtime. Survey resources and their requirements differ by technology stack.

The original technology stack implemented by Oracle Scripting is the JTT technology stack. This is still supported in this release, but deprecated from this point forward. The second is the OA framework technology stack. While at runtime there are some small differences in the way scripts appear in a Web browser based on the technology stack, the main manifestation of these differences is evident when administering survey campaigns and when establishing the survey resources that support them.

The base technology stack assigned to a survey campaign cannot be changed after the survey campaign definition is saved. If you are creating new survey campaigns, use of the new technology stack is strongly recommended.

While survey resources provide the same essential functionality regardless of technology stack, they are implemented dramatically differently. The new framework supports more options and more flexibility. The technology stack determined when defining a survey campaign invokes business rules enforced by the application. For example, if you select JTT as the base technology stack for a survey campaign, no OAF resources will be available to assign to that campaign. If you select OAF as the base technology stack, no JTT resources will be accessible to that campaign.

For all sections of this document in which survey resources are discussed, the text is qualified by which technical stack is to be used at runtime.

See Also

- Detailed Description of Survey Resources
- Creating Survey Resources
- Seeded JSP Survey Resources

1.5.2.9.2 Detailed Description of Survey Resources Survey resources are objects defined as part of a survey campaign that display in a Web browser to users of the Scripting Engine Web interface at runtime. Survey campaigns support four survey resources

(header section, footer section, error page, and final page resources), which include two display types (section resources and page resources).

The two display types appear differently at runtime:

- **Section resources** appear as a section of each HTML page that represents a panel in the script. These do not render in page resource display types.

  The header section appears as a header in each HTML page (panel), just above the panel contents (panel text, graphics, questions, and a Continue button).

  The footer section appears at the foot of each HTML page (panel), immediately below the Continue button.

- **Page resources** display as a separate page at the appropriate time during the execution of a script in a Web browser, and do not represent panels.

  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.

Survey Resources at Runtime

The header and footer section survey resources display, respectively, at the top and bottom of each page of a script executed in a Web browser. The center of the HTML page (below the header section and above the footer section) is where each panel of the script appears. The error page and final page survey resources are full HTML pages. When an error condition occurs while the end user is executing a script, the error page is displayed. After the last panel in a script, the final page is displayed. The full-page error page and final page survey resources do not display the header or footer section resources.

Survey Resource Components

Survey resources are comprised of two components. The first component is the survey resource *definition*. The second component, required for all survey resources except a URL, is the *physical* survey resource file itself.

Survey Resource Definitions Survey resource definitions are created from the Survey Resources tab of the Survey Administration console. When you define a survey resource, you create an object in the database that points to either a physical file (HTML or JavaServer Page, based on technology stack), or a URL (for error page or final page resources, supported for the OAF technology stack only). Survey resources *must* be defined before they can be used in a survey campaign. Once

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

Any survey resource you want to reference when creating a survey campaign *must* be defined *before you create the campaign.* Unlike associating a script with a survey campaign, however, you can change any survey resource associated with an open or active survey campaign after survey campaign creation.

Physical Survey Resources 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 (resulting in an error if it is not).

Physical survey resource files must be created by certified, knowledgeable HTML or JavaServer Page developers. This creation is outside the scope of Oracle Applications.

Business rules for physical survey resources differ based on the following:

- Resource type (HTML file, JSP file, or URL)
- Required location for the physical file (database or APPL_TOP) for runtime execution
- Technology stack of the survey campaign

HTML survey resource files are stored in the applications database. Upload the fully tested HTML files when you define the survey resource. HTML survey resource files are supported for OA framework survey campaigns only.

JSP survey resource files are stored on the applications server, in the $OA_HTML directory of the APPL_TOP. There are no mechanisms in place from Oracle Scripting to upload this file. Consult with a system administrator or the Apache Web server administrator if you do not have physical access to the APPL_TOP. JSP survey resource files are supported for JTT survey campaigns only.

URL survey resources must exist on a Web server accessible to the Apache Web server at runtime, at the absolute location corresponding to the definition (for example, http://xxx.xxx). You cannot define URLs as relative (for example, ../home/index.html). To use this survey resource type, it is assumed that the URL already exists on a Web server. If not, the file must be created and loaded so that the URL can be accessed at runtime. This process is outside the scope of Oracle Scripting; no physical component is typically required for a survey administrator using this resource type. URL survey resource files are supported for OA framework survey campaigns only.

Resource Differences Between Technology Stacks

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 resources are optional.

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 resource in the Footer Section field, to ensure it displays as appropriate at runtime.

For more information, see Survey Resources > Seeded JSP Survey Resources.

OA Framework Survey Resources For survey campaigns defined in the OAF 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 resources are not supported for this technology stack.

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. OAF survey campaigns do not require survey resources to be referenced at runtime, although any resources that are referenced must first be defined.

The physical HTML file corresponding to an OAF survey resource definition is uploaded from the Define Survey Resources page of the Survey Administration console at the time of survey resource definition. In the typical flow of creating and administering survey campaigns, the definition of the survey resource (and the uploading of the physical file to the database) precedes the creation of a survey campaign referencing that resource. However, the survey resource used by a survey campaign can be changed at any time (until canceled or completed).

In contrast, survey resources used for JTT survey campaigns can be defined (including the file name of the physical 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.

Resource Specification Not Required for OAF Survey Campaigns OA Framework survey campaigns do not require survey resources to be associated to execute scripts at runtime. If header section and footer section survey resources are not associated with the survey campaign, then headers or footers will simply not appear in the header or footer sections of the HTML page at runtime. If no specific error page or final page resource is specified, a simple error page or final page will be automatically generated by the application. If defining a static HTML page as an error page for OAF survey campaigns, you can also require error information to display on top of the error page by selecting the **Display Error on Top** box.

Survey Resource Properties

Survey resources have the following properties:

Property	Description
Resource Name	Identifies the survey resource definition.
Resource type	Identifies the function of the resource. Options include HTML File Upload, Deprecated - JSP, and URL For Redirect.
Language	Identifies the language intended to be used to execute in the Scripting Engine Web interface.
URL/File Name	The name of the physical file defined in the application by the survey resource definition. Stored in the database (for OAF survey campaigns) or in the $OA_HTML directory (for JTT survey campaigns). Not a property of URL resource types, which instead have a resource URL.
Last Updated Date	Automatically assigned by the database based on the last time the survey resource record is updated.
Owner	This is the login ID of the person who created the survey resource.
Description	An optional field to contain information describing the survey resource. This property is not displayed in the survey resources list, and can only be viewed from the Survey Resource Details page.

Property	Description
Display Type	Identifies the function of the survey resource at runtime. Two display types include section resources and page resources. Section resources (when defined for a survey campaign) appear in a designated section of each HTML page representing a panel in the script. Page resources are full pages that appear either upon the browser reaching an error condition, or after displaying the last panel in a script. Page resources do not contain panel content and do not display section resources.

The table below summarizes the attributes of survey resources discussed in this section.

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

Guidelines

- The Survey Resource Type column lists the type of survey resource as listed in the Resource Type column in the Survey Administration console UI.
- The Technology Stack Supported column indicates which base technology stack is supported by the designated survey resource type.
- The Section Display Type Supported? column indicates whether the section survey resource type (applicable to header sections and footer sections) is supported by the designated survey resource type.
- The Page Display Type Supported? column indicates whether the page survey resource type (applicable to error pages and final pages) is supported by the designated survey resource type.

Base Technology Stack Options

As of this release, the survey component of Oracle Scripting supports two runtime execution models, based on an older and newer technology stack.

Technology Stack Name	Description in UI	Description	Status	Recommendations
JTT	Deprecated - UI	Oracle CRM Technology Foundation (JTT) technology stack.	Deprecated	Do not use
OAF	OA Framework	Oracle Applications Framework	Current	Use for all new survey campaigns

Oracle Corporation strongly recommends creating all new survey campaigns to execute using the Oracle Applications framework. Benefits include:

- Ability to create and execute survey campaigns without defining resources.
- Ability to redirect users to a specified URL upon error condition or upon visiting the last panel in the script.
- Ability to upload HTML resources to the database from the administrative UI.

JTT survey campaigns are still supported at this time, since survey campaigns previously created using this deprecated technology stack may be required to execute for some time to come. Improvements in the software may not be available in the older technology stack as the product evolves. Additionally, enterprises that continue to create new survey campaigns for execution in the deprecated JTT technology stack risk a future loss of data collected during the course of a survey campaign cycle, when upgrading to a subsequent release in which the deprecated technology stack may become obsolete or non-functional. Since collecting survey data may be the primary purpose for executing a survey campaign, such loss can be catastrophic to the business goals of the enterprise. Oracle Corporation is not responsible for a loss of data caused as a result of remaining with a deprecated technology stack.

See Also

- Survey Resources Differentiated by Technology Stack
- Creating Survey Resources
- Seeded JSP Survey Resources

1.5.2.9.3 Creating Survey Resources Creating OAF Survey Resources

Creating and modifying the physical HTML files that serve as survey resources for OAF 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 resource at the survey campaign level.

Creating JTT Survey Resources

Creating, modifying, and uploading the physical JSP files that serve as survey resources for JTT survey campaigns are not accomplished from within the Survey Administration console.

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.

Creation of the physical files for use as survey resources is outside of the scope of Oracle Applications.

However, as described in the section Test Survey Resources, 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 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 Corporation.

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

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 OAF Survey Resources for Implementation Testing

If you do not associate custom survey resources for a survey campaign using the OAF 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 resources, and utilizing the default page resources, provides a method to verify that survey resources are appropriately displayed upon execution of a survey campaign deployment using the OAF 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 resources displaying, you can assign custom survey resources to an OAF survey campaign and retest.

See Also

- Survey Resources Differentiated by Technology Stack
- Detailed Description of Survey Resources
- Seeded JSP Survey Resources

1.5.2.9.4 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 most likely want to create your own 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

G, and available with any Rapid Install from release 11.5.5 and later. These resources are physically located in $OA_HTML on the applications server.

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 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).

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 resources displaying, you can change the survey resources assigned to a survey campaign to use customized survey resources.

Additional JSP Error Page 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 Resource

Following is the source code for the sample test JSP header section. *This sample is provided for informational purposes only.* Oracle Corporation 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>
```

See Also

- Survey Resources Differentiated by Technology Stack
- Detailed Description of Survey Resources
- Creating Survey Resources

1.5.2.10 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.

See Also

- Survey Administration Console
- The Survey Questionnaire
- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Survey URL
- Concurrent Programs Supporting Survey Operations

1.5.2.11 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 11*i*-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.
    - If this parameter uses the OA.JSP template, the deployment is created as part of a survey campaign with the Oracle Applications Framework 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 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 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 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 resource or URL is displayed, the applications session is maintained, and the user can access any functionality accessible in the hosted application.

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

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 three JSP templates, as described in the table below.

JSP Template	Tech Stack	Authentication	Description
OA.JSP	OAF	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	OAF	For targeted deployments, guest user only.	▪ Does not support menu-based hosting for *targeted* deployments. ▪ Uses applications guest user only.
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.

For More Information

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*.

See Also

- Survey Administration Console
- The Survey Questionnaire

- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Concurrent Programs Supporting Survey Operations

1.5.2.12 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:

- SUBMIT GROUP FM REQUEST FROM IES Concurrent Program
- Summarize Survey Data Concurrent Program
- Update Deployment Status Concurrent Program

References

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

See Also

- Survey Administration Console
- The Survey Questionnaire

- Survey Administration Console View List
- Survey Reports
- Survey Hierarchy, Levels and Objects
- Survey Campaigns
- Cycles
- Deployments
- Survey Resources
- Prototypes
- Survey URL

1.5.2.12.1 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
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
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.
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 Also

- Summarize Survey Data Concurrent Program
- Update Deployment Status Concurrent Program

1.5.2.12.2 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.

See Also

- SUBMIT GROUP FM REQUEST FROM IES Concurrent Program
- Update Deployment Status Concurrent Program

1.5.2.12.3 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 Corporation 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.

See Also

- SUBMIT GROUP FM REQUEST FROM IES Concurrent Program
- Summarize Survey Data Concurrent Program

2

Planning Oracle Scripting Projects

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 11*i*-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.

This section includes the following topics:

2.1 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:

See Also

- Planning Oracle Scripting Survey Campaigns
- Planning Web Scripts

2.1.1 Facets of Scripting-Specific Discovery Process

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

1. Text layer
2. Logic layer
3. Technology layer

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

See Also

- Bringing Together the Layers
- Tools to Aid in Scoping and Discovery
- Oracle Scripting Discovery Data Worksheet
- Oracle Scripting Discovery Checklist Tool

2.1.2 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.

See Also

- Facets of Scripting-Specific Discovery Process
- Tools to Aid in Scoping and Discovery
- Oracle Scripting Discovery Data Worksheet
- Oracle Scripting Discovery Checklist Tool

2.1.3 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, 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, 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.

See Also

- Facets of Scripting-Specific Discovery Process
- Bringing Together the Layers
- Oracle Scripting Discovery Data Worksheet

- Oracle Scripting Discovery Checklist Tool

2.1.4 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(s):

   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 Collections
    - 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 or Oracle Collections, 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.

See Also

- Facets of Scripting-Specific Discovery Process

2.1.5 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
___	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				

Script Author Command Types	Required? (Yes/No)	Resourced? (Yes/No)	Planned? (Yes/No)	Template Exists? (Yes/No)
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 Collections			
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 Oracle Collections
___	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)

CONSULTING TIME ESTIMATE **Build:** **Test:**

See Also

- Facets of Scripting-Specific Discovery Process
- Bringing Together the Layers
- Tools to Aid in Scoping and Discovery
- Oracle Scripting Discovery Data Worksheet

2.2 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.

Survey Business Process Flow

The ten steps depicted in the flowchart are crucial to understanding the administration and use of the Survey component of Oracle Scripting. The process flow steps, as enumerated in Figure 2–1 and described briefly below, are required to plan and execute a survey campaign. The *planning aspects* of each process flow step are addressed in detail in each section described below.

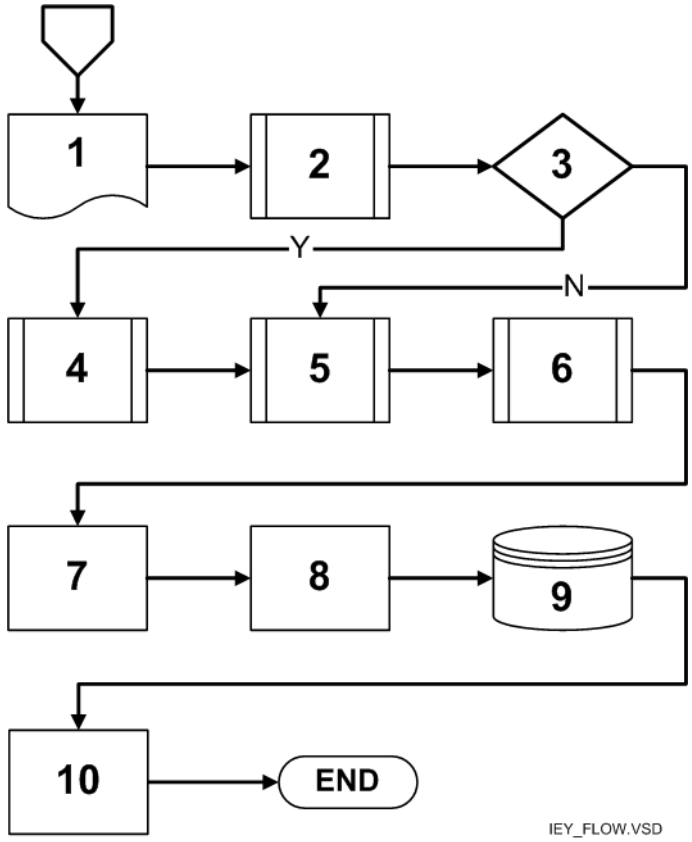


Figure 2–1 Survey Business Process Flow.

1. Define survey campaign requirements.

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.

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?

Are deployments in the survey campaign targeted (list-based), standard, or both? If targeted deployments = NO, skip to step 5. If YES, continue to step 4.

4. Generate lists (for targeted survey campaigns).

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.

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.

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.

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.

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.

As respondents complete survey questionnaires, information is collected in Oracle RDBMS. No action is required to collect this data.

10. Report survey results.

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
- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

See Also

- Planning Agent Interface Projects
- Planning Web Scripts

2.2.1 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.

See Also

- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.1.1 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 Corporation 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.

2.2.1.2 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.

2.2.1.3 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.

2.2.2 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
- Trained Script Developers
- Trained Java, PL/SQL, Oracle Forms, and API Programmers
- Trained Oracle Applications and Database Administrators
- Detailed Script-Specific Information
- Environment-Specific Information
- Detailed Survey Questionnaire Requirements

See Also

- Gathering All Survey Campaign Requirements
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details

- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.2.1 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 JInititator 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 JInititator 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 11*i*-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)

See Also

- Trained Script Developers
- Trained Java, PL/SQL, Oracle Forms, and API Programmers
- Trained Oracle Applications and Database Administrators
- Detailed Script-Specific Information
- Environment-Specific Information
- Detailed Survey Questionnaire Requirements

2.2.2.2 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 below.

See Also

- Appropriate Network, Security, Hardware, and Software Resources
- Trained Java, PL/SQL, Oracle Forms, and API Programmers
- Trained Oracle Applications and Database Administrators
- Detailed Script-Specific Information

- Environment-Specific Information
- Detailed Survey Questionnaire Requirements

2.2.2.3 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.

See Also

- Appropriate Network, Security, Hardware, and Software Resources
- Trained Script Developers
- Trained Oracle Applications and Database Administrators
- Detailed Script-Specific Information
- Environment-Specific Information
- Detailed Survey Questionnaire Requirements

2.2.2.4 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.

See Also

- Appropriate Network, Security, Hardware, and Software Resources
- Trained Script Developers
- Trained Java, PL/SQL, Oracle Forms, and API Programmers
- Detailed Script-Specific Information
- Environment-Specific Information
- Detailed Survey Questionnaire Requirements

2.2.2.5 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).

See Also

- Appropriate Network, Security, Hardware, and Software Resources
- Trained Script Developers
- Trained Java, PL/SQL, Oracle Forms, and API Programmers
- Trained Oracle Applications and Database Administrators
- Environment-Specific Information
- Detailed Survey Questionnaire Requirements

2.2.2.6 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.

See Also

- Appropriate Network, Security, Hardware, and Software Resources
- Trained Script Developers
- Trained Java, PL/SQL, Oracle Forms, and API Programmers
- Trained Oracle Applications and Database Administrators
- Detailed Script-Specific Information
- Detailed Survey Questionnaire Requirements

2.2.2.7 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.

Requirement Type	Description
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.
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.

Requirement Type	Description
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.

2.2.3 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
- List-Based Campaign = YES

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire
- Generating Lists

- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.3.1 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.

See Also

- List-Based Campaign = YES

2.2.3.2 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.

See Also

- List-Based Campaign = NO

2.2.4 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 Online Implementation Guide* for more information on implementing Oracle Marketing. See *Oracle Marketing Online Concepts and Procedures Manual* 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).

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.5 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 Campaigns** 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

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results

- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.6 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

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.7 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 11*i* 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 e-mail 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.

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments

- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.8 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.

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire
- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Collecting Survey Results in Oracle RDBMS
- Reporting Survey Campaign Deployment Results

2.2.9 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.

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire

- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Reporting Survey Campaign Deployment Results

2.2.10 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.*

See Also

- Gathering All Survey Campaign Requirements
- Creating and Deploying a Survey Questionnaire

- Determining If Survey Campaign Requires Targeted Deployments
- Generating Lists
- Administering Survey Resources, Survey Campaign and Cycle Details
- Defining Deployments
- Activating Survey Campaigns
- Monitoring Survey Results
- Collecting Survey Results in Oracle RDBMS

2.3 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.

See Also

- Planning Agent Interface Projects
- Planning Oracle Scripting Survey Campaigns

3

Using the Script Wizard

The Script Wizard feature was introduced to Script Author in Oracle Scripting release 11.5.9 (Interaction Center Family Pack Q). Users of this feature can quickly and easily create simple scripts or surveys by providing script information in a series of windows known as a wizard. The Script Wizard can help reduce the time it takes to train a non-technical Script Author user to create and modify simple scripts. While hiding complexity from non-technical users, the Script Wizard feature makes it possible to re-use questions and answers, reduce keystrokes and data entry errors, and reduce some repetitious work needed to create and modify a script. The Script Wizard is accessible only to users of the Script Author Java applet in an authenticated Oracle Applications session.

This section includes the following topics:

3.1 Getting Started with the Script Wizard

This section includes the following topics:

See Also

- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Panel Manager
- Using the Question Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.1.1 Introduction to Wizard Scripts

Scripts created by the Script Wizard are referred to as *wizard scripts*. In contrast, scripts created using the Script Author's graphical editing tools are now referred to as *graphical scripts*. Regardless of the method used to create a script (graphical tools or Script Wizard), any Script Author script can be executed in any Scripting Engine runtime interface.

All features of wizard scripts are compatible with those of graphical scripts, and wizard scripts can be *graphed* (converted to a copy of a graphical script) for viewing or modification using the graphical tools available to Script Author. At this time, there is no backward compatibility, nor can graphical scripts be converted to wizard scripts. Since the Script Wizard cannot open any graphical script for editing, when you graph a wizard script, the original wizard script is retained.

See Also

- Wizard Script Hierarchy
- Who Uses the Script Wizard?
- What Do Wizard Scripts Contain or Exclude?
- Accessing the Script Wizard

3.1.2 Wizard Script Hierarchy

Wizard scripts are constructed using a certain hierarchy. From top to bottom, this hierarchy is (1) script, (2) panel, (3) question, (4) answer choice. When you create a script, you first define attributes for the global script using the Define Script Properties window. Then, you create the first panel using the Define Panel Information for <Panel Name> window. If that panel requires one or more questions to be defined, then you define the first question using the Question Manager window. Before proceeding further, if the question type accepts answer choices to display at runtime (which is true if the question UI control specified is a radio button, dropdown list, checkbox group or multi-select list), then you define the answer choices using the Answer Manager window. In this way, creating a script using the wizard begins from the top of the hierarchy (the global script) and works down to the bottom (answer choices for a question in a panel).

At the bottom of the hierarchy, complete all work required, and then work your way back up. In other words, when defining a question in a panel, the lowest level of the hierarchy involves defining answer choices. For each question that requires answer choices to be defined, define them all for that question. Then, proceed back up the hierarchy one step to the question level. Are other questions required? If so, define the next question. If it requires answer choices, define them all, and then return to the question level. When you have defined all questions for a panel, step up in the hierarchy to the panel level. Does the script require another panel? Then define another panel, and step back down the hierarchy as low as required.

Each level in the hierarchy includes a button to navigate to the manager window for the next highest level. Each manager window lists the current objects of that type (panels, questions, or answer choices).

From the Panel Manager, Question Manager, and Answer Manager windows, you can view the list of objects of that type, change their order, create new objects, edit the existing objects, copy the defined objects and modify their properties, delete existing objects, or go to the next highest hierarchical level.

There is also a Script Manager window, which only appears when you edit or open existing wizard scripts. From this window you can edit, copy (and subsequently modify), delete, graph, and deploy existing wizard scripts.

By incorporating the concept of managers, script developers can cycle through the various hierarchical levels necessary to create and edit wizard scripts. This provides the flexibility to define objects at any level that can later be modified, enhanced, or deleted. Business rules of wizard scripts are automatically enforced, preventing you from defining an irrelevant object. When you make changes that put defined objects at risk (for example, when changing a question's UI type from a radio button, which requires answer choices, to a text area, which does not), you will be advised accordingly.

The implementation of mangers in the Script Wizard also adds to the script developer's flexibility and decreases the amount of time necessary to create or modify a script. For example, unlike graphical scripts, wizard scripts do not require every panel to contain a question. If a Continue button is required, but no other response is solicited from the script end user, a Continue button is generated automatically. Thus, panels containing only panel text do not require questions to be defined. Similarly, for questions using text question UI controls (text field, text area, or password field), answer choices are not required. In lieu of being prompted for this information, panels with these question types prompt you for question details, allowing you to quickly and easily associate response validation for these questions without requiring a single line of custom code.

3.1.3 Who Uses the Script Wizard?

Skill Set Required

While the Script Wizard makes script development more accessible to non-technical users, script project managers are cautioned that an understanding of business rules and business process flow is still absolutely essential to developing successful scripts using this feature. If Script Wizard users are also the individuals at your enterprise who plan custom scripts, a strong sense of business process engineering and the specific business needs of the script are crucial.

Since wizard scripts are created from Script Author, access to the Scripting Administrator responsibility is also required.

Security and Administrative Users

Users with the Scripting Administrator responsibility have access to Script Author, and can create new scripts or modify existing scripts. Scripts can contain actions

that may include custom Java, PL/SQL, forms, or other commands. While this capability allows valuable functionality, it also poses a potential risk as well. For example, scripts with SQL code can access sensitive database tables. Federal, state, or local regulations may exist regarding access to specific data. Resulting information, if misused, can introduce liability issues for the enterprise.

In addition, these users have access to the Scripting Administration console and can modify, deploy, and delete deployed wizard and graphical scripts, delete custom Java, and change specific Java mappings. If performed inadvertently or carelessly, any of these tasks could cause loss of data or loss of script functionality.

For these reasons, Oracle Corporation strongly recommends that only trusted users be provided with Scripting Administrator responsibility.

See Also

- Introduction to Wizard Scripts
- Wizard Script Hierarchy
- What Do Wizard Scripts Contain or Exclude?
- Accessing the Script Wizard

3.1.4 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?

Properties you will find in any wizard script 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. If you choose not to define any questions in the Question Manager, the sole user interface control for the panel will be the automatically generate Continue button. If you specify one or more questions, the panel at runtime will display each question control for each explicitly defined question, in addition to the Continue button.

- Wizard scripts may include distinct branching.

  This is accomplished by designating a specific panel as the destination for the flow of the script. This option is available at the panel level or the question level. At the panel level, a distinct branch is created when you select **Go to a specific panel (ADVANCED)** as the exit panel sequence. At the answer choice definition level, a distinct branch is created when you select **Go to a specific panel (ADVANCED)** in response to the prompt, "what should happen when the user selects this answer choice?"

- 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 window 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 window. 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 or 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.

See Also

- Introduction to Wizard Scripts
- Wizard Script Hierarchy
- Who Uses the Script Wizard?
- Accessing the Script Wizard

3.1.5 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. (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.

Prerequisites

None

Responsibility

Scripting Administrator

Steps

1. Using an Oracle Applications 11*i*-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.
3. 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.
4. 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.
5. From the Script Author File menu or toolbar, you can access the Script Wizard to create or edit wizard scripts (see References below).

References

- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Opening a Wizard Script from the Script Manager

See Also

- Introduction to Wizard Scripts
- Wizard Script Hierarchy
- Who Uses the Script Wizard?
- What Do Wizard Scripts Contain or Exclude?

3.2 Creating a Wizard Script

Wizard scripts are created using the Script Wizard feature of Script Author. Like graphical scripts, wizard scripts can be executed in any runtime Scripting Engine interface. Unlike graphical scripts, wizard scripts only allow you to save the script when it is syntactically correct. Thus, any saved wizard script can be deployed and executed.

Use this procedure to create a wizard script.

Prerequisites

Script Author must be open. See Accessing the Script Wizard.

Steps

1. If you want to create a new wizard script from the Script Author toolbar:
   a. Click the **Script Wizard** icon.

      The Script Wizard - Choose Create or Edit window appears. Hereinafter, this is referred to as the Choose Create or Edit window.

   b. Select **Create Wizard Script** and click **Next**.

      The Script Wizard - Untitled - Define Script Properties window appears. Hereinafter, this is referred to as the Define Script Properties window.

      In this window, you define global script properties such as the name of the script (recognized by the database) and the script language. Comments are optional.

2. If you want to create a new wizard script from the Script Author menu:
   a. Select **File > New**.

      The New Script window appears.

   b. Click **Wizard script**.

      The Script Wizard - Untitled - Define Script Properties window appears. Hereinafter, this is referred to as the Define Script Properties window.

      In this window, you define global script properties such as the name of the script (recognized by the database) and the script language. Comments are optional.

3. When you reach an appropriate point in the wizard script, save your work.

References

- Accessing the Script Wizard

See Also

- Getting Started with the Script Wizard
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Panel Manager
- Using the Question Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.3 Opening a Wizard Script from Script Author

Wizard scripts are created using the Script Wizard feature of Script Author. You can open a wizard script from the Script Author or from the Script Wizard Script Manager window. To retain the ability to access the script using the Script Wizard, only modify the script using Script Wizard.

Use this procedure to open a wizard script from Script Author.

Prerequisites

Script Author must be open. See Accessing the Script Wizard.

Steps

1. If you want to open an existing wizard script from the Script Author toolbar:
   a. Click the **Script Wizard** icon.

      The Script Wizard - Choose Create or Edit window appears.

   b. Select **Edit Wizard Script** and click **Next**.

      The Script Wizard - Script Manager window appears. Hereinafter, this is referred to as the Script Manager window.

      From this window you can select a script and edit, copy, delete, graph, or deploy an existing wizard script.

2. If you want to open an existing wizard script from the Script Author menu:
   a. Select **File > Open**.

      The Open Script window appears.

   b. Click **Wizard script**.

      The Script Wizard - Script Manager window appears. Hereinafter, this is referred to as the Script Manager window.

      From this window you can select a script and edit, copy, delete, graph, or deploy an existing wizard script.
3. When you reach an appropriate point in the wizard script, save your work.

References

- Opening a Wizard Script from the Script Manager

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Using the Script Manager
- Using the Panel Manager
- Using the Question Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.4 Using the Script Manager

The Script Manager window of the Script Wizard allows you to manage wizard scripts within the wizard interface. From this window you can edit, copy, delete, graph, and deploy existing wizard scripts for your environment.

This section includes the following topics:

- Opening a Wizard Script from the Script Manager
- Editing an Existing Wizard Script

- Copying a Wizard Script
- Deleting a Wizard Script
- Graphing a Wizard Script
- Deploying a Wizard Script
- Defining Wizard Script Properties

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Panel Manager
- Using the Question Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.4.1 Opening a Wizard Script from the Script Manager

Wizard scripts are created using the Script Wizard feature of Script Author. You can open a wizard script from Script Author or from the Script Wizard Script Manager window. To retain the ability to access the script using the Script Wizard, only modify the script using Script Wizard.

Use this procedure to open a wizard script from the Script Manager window of the Script Wizard.

Prerequisites

Script Author must be open. See Accessing the Script Wizard.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:
   - Select **File > Open > Wizard Script**.
   - Click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

The Script Wizard - Script Manager window appears. Hereinafter, this is referred to as the Script Manager window.

2. Navigate through the list of existing wizard scripts, and select the appropriate record.

   The selected record is highlighted.

3. To open the selected wizard script, click **Edit**.

   The selected wizard script opens to the Define Script Properties window.

4. If you make any changes to the script, save your work.

See Also

- Editing an Existing Wizard Script
- Copying a Wizard Script
- Deleting a Wizard Script
- Graphing a Wizard Script
- Deploying a Wizard Script
- Defining Wizard Script Properties

3.4.2 Editing an Existing Wizard Script

Using the Script Wizard, you can open previously created wizard scripts. The list of scripts displayed includes scripts that are deployed from the wizard as well as scripts that were created and saved but not deployed. You cannot open graphical scripts from the wizard, nor can you open wizard scripts that were subsequently graphed and modified as a graphical script. To retain the ability to access the script using the Script Wizard, only modify the script using Script Wizard.

You can view and edit an existing wizard script at any time. If a script is being used as the questionnaire script for an active survey campaign, it is locked. In this case, while the Script 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.

Use this procedure to edit a wizard script from the Script Manager window of the Script Wizard.

Prerequisites

Script Author must be open. See Accessing the Script Wizard.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:
   - Select **File > Open > Wizard Script**.
   - Click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

   The Script Manager window appears.
2. Navigate through the list of existing wizard scripts, and select the appropriate record.

   The selected record is highlighted.
3. To open the selected wizard script, click **Edit**.

   The selected wizard script opens to the Define Script Properties window.
4. Verify the appropriate script is open by looking at the Script Wizard title bar.

   The name of the wizard script is indicated in the title bar. For example, if the selected wizard script is named WS1, the title bar reads Script Wizard - WS1 - Define Script Properties.
5. Optionally, if you want to retain a copy of the original wizard script and save your modifications as a different file, then in the Script Name field, change the name of the current wizard script and any other properties such as the description or language, and click **Next**.

   For example, change WS1 to WS2.
6. Continue editing the wizard script as appropriate.
7. Save your work.

See Also

- Opening a Wizard Script from the Script Manager
- Copying a Wizard Script
- Deleting a Wizard Script
- Graphing a Wizard Script
- Deploying a Wizard Script
- Defining Wizard Script Properties

3.4.3 Copying a Wizard Script

Using the Script Manager window, you can copy a previously created wizard script. You can then rename the copy using the Define Script Properties, and modify the script if required.

Use this procedure to copy a wizard script from the Script Manager window of the Script Wizard.

Prerequisites

Script Author must be open. See Accessing the Script Wizard.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:
   - Select **File > Open > Wizard Script**.
   - Click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

   The Script Manager window appears.
2. Navigate through the list of existing wizard scripts, and select the record for the script you want to copy.

   The selected record is highlighted.
3. To copy the selected wizard script, click **Copy**.

   The selected wizard script is copied. The copied script opens to the Define Script Properties window. The words **Copy of** are prepended to the script name.

   For example, the copy of a script named WS1 is named Copy of WS1.
4. Make any changes to the global properties of the wizard script as appropriate, and then click **Next**.

   For example, add or change the description to match the purpose for the new script.

   The Panel Manager window appears. The **Save** option is now enabled.
5. Optionally, make any other changes to the wizard script as appropriate.
6. Save your work.

See Also

- Opening a Wizard Script from the Script Manager
- Editing an Existing Wizard Script
- Deleting a Wizard Script
- Graphing a Wizard Script
- Deploying a Wizard Script
- Defining Wizard Script Properties

3.4.4 Deleting a Wizard Script

You can delete any unlocked wizard script from the Script Wizard using the Script Manager window. Alternatively, you can view, select, and delete deployed wizard scripts from the Scripting Administration Console. Scripts must be unlocked (they must not be part of an active survey campaign) in order to delete them.

> **Note:** Script deletion is final. Before deleting any script, you may want to back it up on your filing system, network, or on removable media, following your enterprise's security or retention policy. Oracle Corporation is not liable for any scripts that are deleted from your system.

Use this procedure to delete a selected wizard script from the Script Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The script you want to delete must be created with the Script Wizard and must exist in the database as either a saved or deployed script.
- The script must not be locked. If the script is locked, you will first need to unlock the script to delete it.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:
   - Select **File > Open > Wizard Script**.

- Click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

The Script Manager window appears.

2. Navigate through the list of existing wizard scripts, and select the record for the script you want to delete.

   The selected record is highlighted.

3. To remove the selected wizard script from the applications database, click **Delete**.

   The Select Confirmation window appears, warning that you will not be able to view or edit the script if the delete operation is continued.

4. If you want to cancel the delete operation, then from the Delete Confirmation window, click **No**.

   The Script Manager window appears. The script is included in the list of available wizard scripts.

5. If you want to delete the script, then from the Delete Confirmation window, click **Yes**.

   The Script Manager window appears. The script is removed from the database and does not appear in the script manager list.

References

For information on removing a script from the database using the Scripting Information console, refer to *Oracle Scripting Implementation Guide*.

See Also

- Opening a Wizard Script from the Script Manager
- Editing an Existing Wizard Script
- Copying a Wizard Script
- Graphing a Wizard Script
- Deploying a Wizard Script
- Defining Wizard Script Properties

3.4.5 Graphing a Wizard Script

Wizard scripts can only be edited and viewed using the Script Wizard feature of Script Author. To edit or view a script using Script Author graphical tools, you can graph the script, which creates a graphical script copy of the wizard script. The original wizard script is retained with its original file name, and the words **Copy of** are prepended to the script name of the new graphical script.

This section includes the following topics:

- Graphing a Wizard Script from the Script Manager
- Graphing a Wizard Script when Saving

See Also

- Opening a Wizard Script from the Script Manager
- Editing an Existing Wizard Script
- Copying a Wizard Script
- Deleting a Wizard Script
- Deploying a Wizard Script
- Defining Wizard Script Properties

3.4.5.1 Graphing a Wizard Script from the Script Manager

Use this procedure to graph a wizard script from the Script Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The script you want to graph must be created with the Script Wizard and must exist in the database as either a saved or deployed script.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:
   - Select **File > Open > Wizard Script**.
   - Click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

   The Script Manager window appears.

2. Navigate through the list of existing wizard scripts, and select the record for the script you want to graph.

   The selected record is highlighted.

3. To graph the selected wizard script, click **Graph**.

   The Create Graphical Script Copy window appears. This window cautions that graphed scripts can no longer be accessed using the Script Wizard. A copy of the wizard script appears in the Script Author visual layout.

4. If you want to cancel the graph operation, then from the Create Graphical Script Copy window, click **No**.

   The Script Manager window appears. The script is included in the list of available wizard scripts. No graphical copy is created.

5. If you want to graph the script, then from the Create Graphical Script Copy window, click **Yes**.

   The Script Wizard closes. The newly graphed script appears in the visual layout of Script Author. The global script properties include the original name of the wizard script, prepended with the words **Copy of**.

   For example, if the wizard script name is WS1, the resulting graphical script copy is named Copy of WS1.

6. Optionally, using Script Author, make the necessary changes to the graphical script as appropriate.

7. Save your work.

See Also

- Graphing a Wizard Script when Saving

3.4.5.2 Graphing a Wizard Script when Saving

Use this procedure to graph a wizard script from the save sequence of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The wizard script must be syntactically correct and contain at least one panel.

Steps

1. From any point in the Script Wizard that the Save option is enabled, click **Save**.

   The Save Script <script name> Window appears.

2. If you want to save a copy of this script with a different name, select **Yes** and click **Next**.

   - The Save Script <script name> window appears. In the Script Name field, the original name of the script appears, prepended by the words **Copy of**.
   - Make any changes and click **Next**.

   The Continue With Script Save window appears.

3. In the What action do you want to take area, select either **Save and exit** or **Save, deploy and exit** and click **Next**.

   Note: The Save and continue editing option does not provide you with the option to graph a wizard script.

   The Create Graphical Script Copy window appears.

4. If you want to graph the script and immediately view it, then in the Create Graphical Script Copy window, select **Yes** and click **Next**. If you do not want to create a graphical copy, select **No** and click **Next**.

   The Script is Complete for <script name> window appears.

5. Click Finish.

   The Script Wizard closes, and the Script Author Java applet appears.

   If you indicated that you wanted to create a graphical copy of the script, the graphical copy of the wizard script is displayed in the Script Author visual layout. The script view shows the entire graph in the window.

   If you indicated that you did not want to create a graphical copy of the script, the Script Author frame is visible, but no graphical script is open.

- If you selected **Save, deploy and exit** as your save option, Script Author attempts to deploy the script to the applications database. The deployment success or failure message appears in the Script Author status area.

References

- Saving and Deploying Wizard Scripts

See Also

- Graphing a Wizard Script from the Script Manager

3.4.6 Deploying a Wizard Script

The Script Manager window lists all wizard scripts created for your specific environment, including all scripts that were saved but not deployed.

> **Note:** Deploying a script makes it available to execute in the Scripting Engine.

Use this procedure to deploy a wizard script from the Script Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The script must be syntactically correct to deploy. Wizard scripts are syntactically correct when the **Save** option is enabled, or when listed in the Script Manager window.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:
   - Select **File > Open > Wizard Script**.
   - Click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

   The Script Manager window appears.
2. Navigate through the list of existing wizard scripts, and select the record for the script you want to deploy.

   The selected record is highlighted.
3. To deploy the selected wizard script, click **Deploy**.

   The Deploy Script window appears. This window cautions that the selected script will be deployed to the applications database and the Script Wizard will close.

4. If you want to cancel the deploy operation, then from the Deploy Script window, click **No**.

   The Script Manager window appears. The script is included in the list of available wizard scripts, but has not been deployed.

5. If you want to deploy the script, then from the Deploy Script window, click **Yes**.

   Note: If a wizard script with the same name in the Script Name field of the Define Script Properties window 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.

   The Script Wizard closes. The selected script is deployed to the applications database. The script can now be executed in the Scripting Engine agent interface immediately. 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.

See Also

- Opening a Wizard Script from the Script Manager
- Editing an Existing Wizard Script
- Copying a Wizard Script
- Deleting a Wizard Script
- Graphing a Wizard Script
- Defining Wizard Script Properties

3.4.7 Defining Wizard Script Properties

Each wizard script has three configurable global properties. These include the script name, a description of the script, and a language.

The Script Name field contains the name by which a script is identified in the applications database. Oracle Corporation recommends avoiding the inclusion in the script name of special characters typically reserved by database applications or operating systems. For example, avoid the use of slash (/) or backslash (\), colon (:), percent (%), asterisk (*), question mark (?), explanation point (!), tilde (~), or left or

right angle bracket (< or >) characters. The script name property can include dash and underscore characters as well as space characters if desired.

The Description field is the only optional script property. This text area is intended for script developers to provide some background or helpful information regarding the purpose of the script or details of its construction. It is visible to anyone who may open the script to view or modify it.

The Language field reads and displays available languages from the Oracle RDBMS FND_LANGUAGES table. Select the appropriate language for your environment, if it is not already displayed. For example, for English in the United States, use the default AMERICAN setting.

These properties equate to the global script properties accessed in a graphical script from 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 (and then deploy the script).

Use this procedure to define the configurable global properties for a wizard script.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- A wizard script must exist for which you want to define global properties.

Steps

1. Start the Script Wizard from Script Author using one of the following methods:

   a. If you want to access the script properties of an existing script:

   - Select **File > Open > Wizard Script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

     The Script Manager window appears.

   - Select the appropriate record and click **Edit**.

     The Define Script Properties window appears.

   b. If you want to access the script properties of a new wizard script:

   - Select **File > New > Wizard script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Create Wizard Script**, and click **Next**.

The Define Script Properties window appears.

2. In the Define Script Properties window, in the Script Name field, type the name you want this script to be called.
3. Optionally, in the Description field, describe the script, its purpose, intended function, developers, or other pertinent information.
4. In the Language field, select the appropriate language for your environment, if it is not already displayed.

   For example, for English in the United States, use the default AMERICAN setting.
5. When you are satisfied with the settings for the current wizard script's global properties, click **Next**.

   The Panel Manager window appears.

   If editing an existing, syntactically script, the Save option is now enabled.
6. If editing an existing script, save your work.
7. If creating a new script, enter panel information for the first panel in the script and click **Next**.

   - You cannot save a new script until you define at least one panel, and provide the Scripting Engine with instructions on how the script flows at runtime.
   - You can determine flow instructions by specifying a choice in the panel's exit panel sequence.

   The Question Manager window appears. The Save option is now enabled.
8. If creating a new script, save your work.

See Also

- Opening a Wizard Script from the Script Manager
- Editing an Existing Wizard Script
- Copying a Wizard Script
- Deleting a Wizard Script
- Graphing a Wizard Script
- Deploying a Wizard Script

3.5 Using the Panel Manager

The Panel Manager window of the Script Wizard allows you to view and manage existing panels in your wizard script. From this window you can reorder existing panels in your wizard script, as well as create, edit, copy, and delete panels.

This section includes the following topics:

- Accessing the Panel Manager
- Reordering Panel Sequence
- Creating Panels with the Panel Manager
- Editing Existing Panels
- Copying Existing Panels
- Deleting Existing Panels

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Question Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.5.1 Accessing the Panel Manager

The panel manager can only be accessed after a script contains at least one panel. For existing scripts with one or more panels, it is accessed immediately after viewing the wizard script's global script properties.

For new scripts, you must first the wizard script's global script properties, and then define the first panel. Thereafter, you can either continue to define a panel's properties (including any questions and answer choices), or you can then access the panel manager, and later return to any created panel (from the panel manager) and modify its properties, questions, and answer choices, as appropriate.

Use this procedure to access the Panel Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- To access the panel manager, at least one panel must already be defined in the wizard script.

Steps

Start the Script Wizard from Script Author using one of the following methods:

1. If you want to access the panel manager of an existing script:
   a. Select **File > Open > Wizard Script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

      The Script Manager window appears.

   b. Select the record of the wizard script for which you want to access the panel manager, and click **Edit**.

      The Define Script Properties window appears.

   c. Click **Next**.

      The Script Wizard - <script name> - Panel Manager window appears. Hereinafter, this is referred to as the Panel Manager window.

2. If you want to access the Panel Manager window of a new wizard script:
   a. Select **File > New > Wizard script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Create Wizard Script**, and click **Next**.

      The Define Script Properties window appears.

   b. Define properties for the wizard script and click **Next**. For more information, see Defining Wizard Script Properties.

      The Script Wizard - <script name> - Define Panel Information for Panel <panel name> appears. Hereinafter, this is referred to as the Define Panel Information window.

   c. Define panel details such as the panel name, panel text, text style, and the desired alignment of the panel text and question. For more information, see Creating Panels with the Panel Manager.

Alternatively, you can enter temporary information in this window, and change it later by accessing the panel from the panel manager. For more information, see Editing Existing Panels.

d. Optionally, if this panel requires a question to be defined, enter question details. For more information, see Using the Question Manager.

e. Optionally, if your panel contains one or more questions and requires answer choices to be defined, define the answer choices. For more information, Using the Answer Manager.

f. When panel details are complete, save the wizard script and continue editing. For more information, see Saving and Deploying Wizard Scripts.

The Script Wizard - <script name> - Save and Continue Editing Script window appears. Hereinafter, this is referred to as Save and Continue Editing Script window.

g. From the Save and Continue Editing Script window, click **Next**.

The Script Wizard - <script name> - Panel Manager window appears. Hereinafter, this is referred to as the Panel Manager window.

See Also

- Reordering Panel Sequence
- Creating Panels with the Panel Manager
- Editing Existing Panels
- Copying Existing Panels
- Deleting Existing Panels

3.5.2 Reordering Panel Sequence

The Panel Manager window lists all panels currently defined in your wizard script. Panels are listed in the sequence in which they are created. When two or more panels are defined in a wizard script, you can reorder panel sequence. 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. If the exit panel sequence specify a designated panel or the end of the script, recording panel sequence will have no effect.

Use this procedure to change the order of panels in the flow of a wizard script from the Panel Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- Two or more panels must be defined in the wizard script.
- The order of the panel in the existing sequence must support the ability to move it up or down in the script sequence.

  For example, you cannot move the first panel any higher in the sequence of panels in the script, since it will appear first. Also, you cannot move the last panel any lower down, since it will appear as the last panel in the flow of the script.

Steps

1. Access the panel manager for the appropriate script.
2. If you want to move a panel to be higher in the flow of the script (to be displayed at runtime sooner than the order in which it was created), perform the following:

   a. In the Panel Manager window, select the record for the appropriate panel.

   For example, assume the wizard script contains the following three panels in sequence: Default Panel 1, Panel2, Panel3. Select the record for Panel3.

   Note: If a selected panel can be promoted to a higher level, the **Move Up** option is enabled. If this option is not enabled, you cannot change this panel to be higher in the script sequence.

   b. Click **Move Up** once.

   The panel moves one sequential number higher in the panel manager, indicating it will now display at runtime sooner (if the business logic of the script supports the display of the panel).

   c. Repeat the previous step for each level higher that you want to promote the panel.

   For example, move the panel up one more level. The sequence of panels for this example is now Panel3, Default Panel 1, Panel2.

3. If you want to move a panel to be lower down in the flow of the script (to be displayed at runtime later than the order in which it was created), perform the following:

a. In the Panel Manager window, select the record for the appropriate panel.

   For example, assume the wizard script contains the following three panels in sequence: Panel A, Panel B, and Panel C. Select the record for Panel A.

   Note: If a selected panel can be demoted to a lower level, the Move Down option is enabled. If this option is not enabled, you cannot change this panel to be lower in the script sequence.

b. Click **Move Down** once.

   The panel moves one sequential number lower in the panel manager, indicating it will now display at runtime one panel later (if the business logic of the script supports the display of the panel).

c. Repeat the previous step for each level lower that you want to demote the panel.

   For example, move the panel down one more level. The sequence of panels for this example is now Panel B, Panel C, Panel A.

4. Save your work.

See Also

- Accessing the Panel Manager
- Creating Panels with the Panel Manager
- Editing Existing Panels
- Copying Existing Panels
- Deleting Existing Panels

3.5.3 Creating Panels with the Panel Manager

Each wizard script requires one or more panels to be defined before it can be executed using the Scripting Engine. When creating a new wizard script, you are automatically prompted for details for the fist panel. Subsequently, you can create new panels from the panel manager. If saving and continuing to edit a script, you are also routed to the panel manger by the Script Wizard.

Use this procedure to create panels from the Panel Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script in order to access the panel manager.

Steps

1. Access the panel manager for the appropriate script.
2. To create a new panel, click **Create**.

   The Script Wizard - <script name> - Define Panel Information for Panel <panel name> appears. Hereinafter, this is referred to as the Define Panel Information window.
3. In the Panel Name field, type a unique panel name.

   Each panel on the same graph of a script must have a unique panel name. All panels created in a wizard script are on the same graph (the root graph).
4. In the Panel Text field, overwrite the default message with your own text. If no text is required, delete the words **Enter text here.**

   In some cases, question labels provide enough information and no panel text is required.
5. From the Panel Text Style list, select one of the following options:
   a. If you want to highlight the text in the Panel Text field as text an agent should speak or as primary information to relay in the panel, select **Spoken Text**.
   b. If you want to indicate that the text in the Panel Text field provides instructions to the script end user, select **Instructions**.
   c. If you do not wish to apply any specific formatting, select **Default**.
6. From the Text Alignment list, select the default formatting for all paragraphs of panel text.
7. From the Question Alignment list, select the default formatting for all questions included in the panel.
8. Select an exit panel sequence based on the following information:
   a. If you want the script to flow to the next panel to be created in the panel manager, select **Go to the next panel in sequence**.

b. If you want the script to progress to a specific panel, select **Go to a specific panel (ADVANCED)**.

c. If you want the script to terminate after exiting the current panel, select **End Script**.

9. When satisfied with the information provided for the current panel, click **Next**.
10. Save your work.

See Also

- Accessing the Panel Manager
- Reordering Panel Sequence
- Editing Existing Panels
- Copying Existing Panels
- Deleting Existing Panels

3.5.4 Editing Existing Panels

Once defined, you can modify an existing panel by selecting it from the Panel Manager window.

Use this procedure to edit an existing panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. Access the panel manager for the appropriate script.
2. In the Panel Manager window, select the record for the appropriate panel and click **Edit**.

   The Define Panel Information window appears.
3. To modify panel information appearing in this window, make the appropriate changes and click **Next**.

   The Question Manager window appears. If you do not define a question for a panel in the Script Wizard, the panel is provided with an automatically

generated Continue button that directs the script to the appropriate destination based on the panel's exit panel sequence.

4. To exit the Question Manager window without defining any questions, from the Question Manager window, select Panel Manager.
5. If you want to reorder, edit, copy, or delete existing questions for this panel, or create a new question for the panel, then in the Question Manager window, select the appropriate control.

   For more information, see Using the Question Manager.
6. If you want to reorder, edit, copy, or delete existing answer choices for a radio button, dropdown list, checkbox group, or multi-select list question, or create a new answer choice for any of the listed question types, then in the Answer Manager window, select the appropriate control.

   For more information, see Using the Answer Manager.
7. Save your work.

See Also

- Accessing the Panel Manager
- Reordering Panel Sequence
- Creating Panels with the Panel Manager
- Copying Existing Panels
- Deleting Existing Panels

3.5.5 Copying Existing Panels

You can copy any existing panel in a wizard script from the Panel Manager window. The copied panel contains the exact same characteristics as the original panel, with the exception that the panel name of the new panel is prepended with the words **Copy of** (to ensure uniqueness of panel names). Thereafter, you can modify panel details as appropriate.

Use this procedure to copy an existing panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. Access the panel manager for the appropriate script.
2. In the Panel Manager window, select the record for the appropriate panel and click **Copy**.

   The Define Panel Information window appears for the newly copied panel. The panel name is prepended with Copy of.

   For example, if the original panel is named Original, then the copied panel name is Copy of Original.
3. Optionally, modify panel information appearing in the Define Panel Information window, and click **Next**.
4. Save your work.

See Also

- Accessing the Panel Manager
- Reordering Panel Sequence
- Creating Panels with the Panel Manager
- Editing Existing Panels
- Deleting Existing Panels

3.5.6 Deleting Existing Panels

You can delete any existing panel in a wizard script from the Panel Manager window.

Note: Deleting a panel permanently removes the panel from the script, and automatically reroutes the flow of the script to the next panel in sequence. 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.

Note: 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.

Use this procedure to delete an existing panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. Access the panel manager for the appropriate script.
2. In the Panel Manager window, select the record for the appropriate panel and click **Delete**.
3. If the panel you selected is the only panel in the wizard script, then the Script Must Have At Least One Panel window appears, warning that if you delete the last panel, you must create a new one before the wizard script can be saved.
   a. If you want to cancel the delete operation, then from the Script Must Have At Least One Panel window, click **No**.

      The Panel Manager window appears. The panel is retained in the list of panels and is not deleted.

   b. If you want to continue the delete operation, then from the Script Must Have At Least One Panel window, click **Yes**.
4. The Deleting This Panel Can Affect Branching window appears, warning that the delete action will affect the panel and all of its components.
   a. If you want to cancel the delete operation, then from the Deleting This Panel Can Affect Branching window, click **No**.

      The Panel Manager window appears. The panel is retained in the list of panels and is not deleted.

   b. To permanently remove the selected panel from the wizard script, click **Yes**.

      The Panel Manager window appears. The deleted panel is no longer included in the list of panels.
5. Save your work.

See Also

- Accessing the Panel Manager
- Reordering Panel Sequence
- Creating Panels with the Panel Manager
- Editing Existing Panels
- Copying Existing Panels

3.6 Using the Question Manager

The Question Manager window of the Script Wizard allows you to view and manage panel questions in your wizard script. From this window you can reorder existing questions within a selected panel in your wizard script, as well as create, edit, copy, and delete questions.

This section includes the following topics:

- Accessing the Question Manager
- Reordering Question Sequence
- Creating Questions with the Question Manager
- Editing Existing Questions
- Copying Existing Questions
- Deleting Existing Questions

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Panel Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.6.1 Accessing the Question Manager

Use this procedure to access the Panel Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- At least one panel must already be defined in the wizard script.
- The Question Manager window applies only to the questions in that panel. To access each question manager, you must access the designated panel.

Steps

Start the Script Wizard from Script Author using one of the following methods:

1. If you want to access the Question Manager window of an existing script:
   a. Select **File > Open > Wizard Script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

      The Script Manager window appears.

   b. Select the record of the wizard script that you want to open, and click **Edit**.

      The Define Script Properties window appears.

   c. Click **Next**.

      The Panel Manager window appears.

   d. Select the record of the panel in this wizard script that you want to open, and click **Edit**.

      The define panel information window appears.

   e. Click **Next**.

      The Script Wizard - <script name> - Question Manager for Panel <panel name> window appears. Hereinafter, this is referred to as the Question Manager window.

2. If you want to access the Question Manager window of a new wizard script:
   a. Select **File > New > Wizard script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Create Wizard Script**, and click **Next**.

      The Define Script Properties window appears.

b. Define properties for the wizard script and click **Next**. For more information, see Defining Wizard Script Properties.

The Panel Manager window appears.

c. Define panel properties for the first panel and click **Next**.For more information, see Creating Panels with the Panel Manager.

The Script Wizard - <script name> - Question Manager for Panel <panel name> window appears. Hereinafter, this is referred to as the Question Manager window.

See Also

- Reordering Question Sequence
- Creating Questions with the Question Manager
- Editing Existing Questions
- Copying Existing Questions
- Deleting Existing Questions

3.6.2 Reordering Question Sequence

The Question Manager window lists all questions currently defined for the selected panel in your wizard script. Questions are listed in the sequence in which they are created. When two or more questions are defined in a panel, you can reorder question sequence to change the appearance of the panel at script runtime.

Use this procedure to change the order of questions in a panel in a wizard script from the Question Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- At least one panel must be defined for the wizard script.
- Two or more questions must be defined for the panel in order to reorder question sequence in the panel.
- The order of the selected question in the existing panel must support the ability to move it up or down in panel question sequence.

  For example, you cannot move the first question any higher in the sequence of questions in a panel, since it will appear first. Also, you cannot move the last

question any lower down, since it will appear as the last question in the panel at runtime.

Steps

1. Access the Question Manager window for a specific question in a wizard script.
2. If you want to move a panel question to be higher in the sequence of questions in the panel, perform the following:
   a. In the Question Manager window, select the record for the appropriate question.

      For example, assume the panel contains three questions in sequence: Question 1, Question 2, Question 3. Select the record for Question 3.

      Note: If a selected question can be promoted to a higher level, the Move Up option is enabled. If this option is not enabled, you cannot change this question to be higher in the panel question sequence.

   b. Click **Move Up** once.

      The question moves one sequential number higher in the Question Manager window, indicating it will now display within the panel sooner (if the business logic of the script supports the display of the panel).

   c. Repeat the previous step for each level higher that you want to promote the question.

      For example, move the question up one more level. The sequence of panels for this example is now Question 3, Question 1, Question 2.
3. If you want to move a question to be lower down in the sequence of questions within the panel (to be displayed at runtime lower in the panel than the order in which it was created), perform the following:
   a. In the Question Manager window, select the record for the appropriate question.

      For example, assume the panel contains the following three questions in sequence: Question A, Question B, Question C. Select the record for Question A.

Note: If a selected question can be demoted to a lower level in the panel, the Move Down option is enabled. If this option is not enabled, you cannot change the sequence of this question to appear lower in the panel sequence.

b. Click **Move Down** once.

   The question moves one sequential number lower in the Question Manager window, indicating it will now display at runtime one panel later (if the business logic of the script supports the display of the panel).

c. Repeat the previous step for each level lower that you want to demote the question.

   For example, move the question down one more level. The sequence of questions for this panel is now Question B, Question C, Question A.

4. Save your work.

See Also

- Accessing the Question Manager
- Creating Questions with the Question Manager
- Editing Existing Questions
- Copying Existing Questions
- Deleting Existing Questions

3.6.3 Creating Questions with the Question Manager

Each script includes at least one question control at runtime. For wizard scripts, if the control required for a panel is only a single Continue button, it will be generated automatically for the panel, and no questions are required to be created for the panel. If the panel requires any other question control supported by the Script Wizard (text field, text area, set of radio buttons, a drop-down list, checkbox group, multi-select list, or a password field), you must define a question using the Question Manager window of the Script Wizard. Panels can have any number of questions, limited only by practicality for the end user.

Use this procedure to create questions from the Question Manager window for any designated panel from the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. Access the Question Manager window for a specific question in a wizard script.
2. In the Question Manager window, click **Create**.

   The Script Wizard - <script name> - Define Question Main Properties window appears. Hereinafter, this is referred to as the Define Question Main Properties window.
3. Question main properties include the question name, the question label, and the question user interface control type. Define the main properties for this question as described below:

   a. In the Question Name field, type the name for this question.

   * This required field indicates the name of the question as stored in the applications database.
   * Limit question names to alphanumeric characters, spaces, underscores and dashes.

   b. Optionally, in the Question Label field, type the label for this question.

   * Limit question labels to alphanumeric characters, spaces, underscores and dashes.
   * Question Label: If executing the script in the Scripting Engine agent interface, the question label appears as the prompt for the specific question.
   * If using reporting, the question label is displayed on reports to identify panel questions and responses.

   c. From the Type list, select the question user interface control type you want to display in the panel at runtime for this question.

   d. Click **Next**.
4. If you designated a type that takes a predefined list of answer choices or lookup values (radio button, drop-down list, checkbox group, or multi-select list), the Answer Manager window appears. These question types require you to identify answer choices to display for the user to select at runtime. Create an answer choice as described in Creating Answer Choices with the Answer Manager.

5. If you selected a text question type (text field, text area, or password field), the Define Question Detail window appears. Define the question details as described below:

   a. If you require boolean response validation (to force the end user of the script to provide a response for this question at runtime), in the **Is this answer mandatory** area, select **Yes**. If you do not require a response for this question to be provided at runtime. select **No**.

   b. If you want to provide a default value for this question, in the Default Value field, type the suggested value.

      * This value will populate the selected question control as a suggested response at runtime.

      * This value can be overwritten at runtime by the user.

      * In a graphical script, this is referred to as passing a response value as a constant, using a constant command.

   c. If you want to provide response validation for this question at runtime, from the Validation list, select the appropriate validation type as indicated below:

      To validate that any response entered is an integer, select **Integer Only**.

      To validate that any response entered is an integer within a specific range, select **Integer Range**, and in the Integer Range area, enter the lowest and highest values for the desired range.

      To validate that any response entered is a valid date (but could be a date in the past), select **Date - Valid Date**.

      To validate that any response entered is a valid date which is later than the current SYSDATE, select **Date - Not in the Past**.

   d. Optionally, click **Next** to return to the Question Manager.

   e. Save your work.

See Also

- Accessing the Question Manager
- Reordering Question Sequence
- Editing Existing Questions
- Copying Existing Questions

- Deleting Existing Questions

3.6.4 Editing Existing Questions

Once defined, you can modify an existing question by selecting it from the Question Manager window.

Use this procedure to edit an existing question for a panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.
- One or more questions must be defined for the specified panel.

Steps

1. Access the Question Manager window for a specific question in a wizard script.
2. In the Question Manager window, select the record for the appropriate question and click **Edit**.

   The Define Question Main Properties window appears.
3. To modify the main properties for a question in this window, make the appropriate changes and click **Next**.

   Question main properties include the question name, the question label, and the question user interface control type. For more information, see Creating Questions with the Question Manager.

   The Question Manager window appears. If you do not define a question for a panel in the Script Wizard, the panel is provided with an automatically generated Continue button that directs the script to the appropriate destination based on the panel's exit panel sequence.

Note: If you modify the question type from one that contains answer choices (radio button, drop-down 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.

Note: If you modify the question type from one that allows branching in the script based on answer choices (radio button or drop-down 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.

4. To exit the Question Manager window without defining any questions, from the Question Manager window, select Panel Manager.
5. Save your work.

See Also

- Accessing the Question Manager
- Reordering Question Sequence
- Creating Questions with the Question Manager
- Copying Existing Questions
- Deleting Existing Questions

3.6.5 Copying Existing Questions

You can copy any existing question defined for a panel from the Question Manager window. The copied question contains the exact same characteristics as the original question, with two exceptions:

1. The name of the new question is prepended with the words **Copy of** (to ensure uniqueness of question names).
2. 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. The matter is determined based on your response to the Only 1 Question Can Determine Branching for this Panel window.

After copying a question, you can modify question details as appropriate. If doing so has specific repercussions, you will see appropriate warning windows.

Use this procedure to copy an existing panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.
- One or more questions must be defined for the specified panel.

Steps

1. Access the Question Manager window for a specific question in a wizard script.
2. In the Question Manager window, select the record for the appropriate question and click **Copy**.

   The Define Question Main Properties window appears. The properties of the original question appear identically, with the words **Copy of** prepended to the original value in the Question Name field.
3. To modify the main properties for a question in this window, make the appropriate changes and click **Next**.
4. If the question is of a text type (text field, text area, or password field), the Define Question Detail window appears. To modify the question detail properties for a question in this window, make the appropriate changes and click **Next**. For more information, see Creating Questions with the Question Manager.
5. If the question type takes a predefined list of answer choices or lookup values (radio button, drop-down list, checkbox group, or multi-select list), the Answer Manager window appears. For more information, see Using the Answer Manager.
6. If you designated a type takes a predefined list of answer choices or lookup values (radio button, drop-down list, checkbox group, or multi-select list), the Answer Manager window appears.
7. Save your work.

See Also

- Accessing the Question Manager
- Reordering Question Sequence
- Creating Questions with the Question Manager
- Editing Existing Questions
- Deleting Existing Questions

3.6.6 Deleting Existing Questions

You can delete any existing question in a wizard script panel from the Question Manager window.

Note: Deleting a question permanently removes any components associated with the question, including answer choices for the question, and branching associated with specific answer choices.

Note: 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* (the panel is no longer displayed or evaluated for any session of the script at runtime).

Use this procedure to delete an existing question.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.
- One or more questions must be defined for the specified panel.

Steps

1. Access the Question Manager window for a specific question in a wizard script.
2. In the Question Manager window, select the record for the appropriate question and click **Delete**.

   The Deleting This Question Can Affect Branching window appears, indicating the repercussions of deleting the question.

3. If you want to cancel the delete operation, then from the Deleting This Question Can Affect Branching window, click **No**.

   The Question Manager window appears. The question is included in the list of available panel questions.

4. If you want to delete the question from this panel, then from the Deleting This Question Can Affect Branching window, click **Yes**.

   The Script Manager window appears. The question is removed from the panel and does not appear in list of panel questions.

5. Save your work.

See Also

- Accessing the Question Manager
- Reordering Question Sequence
- Creating Questions with the Question Manager
- Editing Existing Questions
- Copying Existing Questions

3.7 Using the Answer Manager

The Answer Manager window 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, drop-down lists, checkbox groups, and multi-select lists. For these question types, the Script Wizard routes you to the Answer Manager window automatically when creating a question. From this window you can reorder existing answer choices within a selected question, as well as create, edit, copy, and delete answer choices.

Optionally, for some question types, you can define response validation.

Note: The answer manager does not allow you to 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). It 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. As always, once you modify a wizard script using graphical tools, you cannot view, edit, or deploy the modified script from the Script Wizard

This section includes the following topics:

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Panel Manager
- Using the Question Manager
- Determining Flow with the Script Wizard
- Saving and Deploying Wizard Scripts

3.7.1 Accessing the Answer Manager

Use this procedure to access the Answer Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- At least one panel must already be defined in the wizard script.
- At least one question supporting answer choices must already be defined in the wizard script. These question types include radio buttons, drop-down lists, checkbox groups, and multi-select lists.
- The Answer Manager window applies only to the answer choices for a specific question in a specific panel. To access each Answer Manager window, you must first access the designated panel and question.

Steps

Start the Script Wizard from Script Author using one of the following methods:

1. If you want to access the Answer Manager window for a panel question in an existing script:
   a. Select **File > Open > Wizard Script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Edit Wizard Script**, and click **Next**.

      The Script Manager window appears.

   b. Select the record of the wizard script that you want to open, and click **Edit**.

      The Define Script Properties window appears.

   c. Click **Next**.

      The Panel Manager window appears.

   d. Select the record of the panel in this wizard script that you want to open, and click **Edit**.

      The Define Panel Information window appears.

   e. Click **Next**.

      The Question Manager window appears.

   f. Select the record of the question in this panel for which you want to access the Answer Manager window, and click **Edit**.

      The Define Question Main Properties window appears.

   g. Click **Next**.

The Script Wizard - <script name> - Answer Manager for Question <question name> window appears. Hereinafter, this is referred to as the Answer Manager window.

2. If you want to access the Answer Manager window for a panel question of a new wizard script:

   a. Select **File > New > Wizard script**, or click the **Script Wizard** tool on the Script Author toolbar, select **Create Wizard Script**, and click **Next**.

      The Define Script Properties window appears.

   b. Define properties for the wizard script and click **Next**. For more information, see Defining Wizard Script Properties.

      The Panel Manager window appears.

   c. Define panel properties for the first panel and click **Next**. For more information, see Defining Wizard Script Properties.

      The Question Manager window appears.

   d. Define question properties for this panel, using a question type (radio button, drop-down list, checkbox group or multi-select list) that supports answer choices, and click **Next**. For more information, see Creating Questions with the Question Manager.

      The Script Wizard - <script name> - Answer Manager for Question <question name> window appears. Hereinafter, this is referred to as the Answer Manager window.

See Also

- Reordering Answer Choice Sequence
- Creating Answer Choices with the Answer Manager
- Editing Existing Answer Choices
- Copying Existing Answer Choices
- Deleting Existing Answer Choices

3.7.2 Reordering Answer Choice Sequence

The Question Manager window lists all questions currently defined for the selected panel in your wizard script. Questions are listed in the sequence in which they are

created. When two or more questions are defined in a panel, you can reorder question sequence to change the appearance of the panel at script runtime.

Use this procedure to change the order of questions in a panel in a wizard script from the Question Manager window of the Script Wizard.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- At least one panel must be defined for the wizard script.
- At least one question must be defined for the panel.
- Two or more answer choices must be defined for the question in order to reorder answer choice sequence in the question.
- The order of the selected answer choices in the question must support the ability to move it up or down in sequence with the other answer choices for the question.

  For example, you cannot move the first answer choice for a question any higher in the sequence of answer choices for that question, since it will appear first. Also, you cannot move the last answer choice any lower down, since it will appear as the last answer choice for that question at runtime.

Steps

1. Access the Answer Manager window for a specific question in a wizard script.
2. If you want to move an answer choice to be higher in the sequence of answer choices for that question, perform the following:
   a. In the Answer Manager window, select the record for the appropriate answer choice.

      For example, assume the panel contains three answer choices in sequence: Answer Choice 1, Answer Choice 2, Answer Choice 3. Select the record for Answer Choice 3.

      Note: If a selected answer choice can be promoted to a higher level, the Move Up option is enabled. If this option is not enabled, you cannot change this answer choice to be higher in sequence of available answer choices.

      ---

   b. Click **Move Up** once.

The answer choice moves one sequential number higher in the Answer Manager window, indicating that it will now display within the question sooner (if the business logic of the script supports the display of the panel).

c. Repeat the previous step for each level higher that you want to promote the answer choice.

For example, move the answer choice up one more level. The sequence of panels for this example is now Answer Choice 3, Answer Choice 1, Answer Choice 2.

3. If you want to move an answer choice to be lower down in the sequence of answer choices in the question (to be displayed at runtime lower in the question than the order in which it was created), perform the following:

a. In the Answer Manager window, select the record for the appropriate answer choice.

For example, assume the panel contains the following three answer choices in sequence: Answer Choice A, Answer Choice B, Answer Choice C. Select the record for Answer Choice A.

Note: If a selected answer choice can be demoted to a lower level in the question, the Move Down option is enabled. If this option is not enabled, you cannot change the sequence of this answer choice to appear lower in sequence of available answer choices.

---

b. Click **Move Down** once.

The answer choice moves one sequential number lower in the Answer Manager window, indicating it will now display at runtime one answer choice later (if the business logic of the script supports the display of the panel).

c. Repeat the previous step for each level lower that you want to demote the answer choice.

For example, move the answer choice down one more level. The sequence of answer choices for this panel is now Answer Choice B, Answer Choice C, Answer Choice A.

4. Save your work.

See Also

- Accessing the Answer Manager
- Creating Answer Choices with the Answer Manager
- Editing Existing Answer Choices
- Copying Existing Answer Choices
- Deleting Existing Answer Choices

3.7.3 Creating Answer Choices with the Answer Manager

Panel questions that display text controls (text fields, text areas, and password fields) do not require answer choices to be defined. When defining any other question type, the Script Wizard routes you to the Answer Manager window, where you can define the values which automatically populate the question control at runtime. Answer choices apply only to a specific question for a specific panel.

The answer manager displays different information, based on how many of the defined answer choices can be selected at runtime. Radio buttons and drop-down lists at runtime display all the answer choices programmed for the question, but only allow a single answer choice to be selected. These also only allow a single answer choice to be designated at the default answer choice at runtime.

Checkbox groups and multi-select lists allow none, one, or any combination of the defined answer choices to be selected at runtime. These also support establishing multiple answer choices to be set as default answers at runtime.

Answer choice properties for all question types include:

- The answer value (which is stored in the database).
- The answer label (which is displayed to the script end user at runtime).
- The boolean property designating whether this answer choice is displayed as the default option for this question at runtime.

  For checkbox groups and multi-select lists, more than one answer choice can be selected as a default.

Answer choice properties for radio button and drop-down lists include a column indicating branching specifically associated with selection of that answer choice at runtime. Branching instructions provided at this level override branching instructions provided at the panel level. (The Branching to column in the answer choice properties table is null unless specific branching instructions are associated with an answer choice.) Answer choice-level branching uses distinct branching,

which is accessible to radio button and drop-down list question types since these only support selection of a single answer choice at runtime. Because you cannot tell a script to branch to two or more *different* panels as the next destination, you cannot define answer choice-level branching for checkbox group and multi-select list question types (since these can accept more than one answer choice at runtime). For this reason, the property designating answer choice level branching is not a feature of the Answer Manager window for these question types.

Only a single question in a panel can determine branching based on the answer choices for that panel.

To be syntactically correct, each question that accepts answer choices requires at least one answer choice to be defined. Practically speaking, you will typically define at least two answer choices. There are no limit to the amount of answer choices that can be assigned to a question. Thus, the upward limit tends to be based on the combination of business requirements, as well as considerations of practical usability for the script end user.

Use this procedure to create answer choices from the Answer Manager window for a question in a wizard script panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- At least one panel must already be defined in the wizard script.
- At least one question supporting answer choices must already be defined in the wizard script. These question types include radio buttons, drop-down lists, checkbox groups, and multi-select lists.
- The Answer Manager window applies only to the answer choices for a specific question in a specific panel. To access each Answer Manager window, you must first access the designated panel and question.

Steps

1. Access the Answer Manager window for a specific question in a wizard script.
2. In the Answer Manager window, click **Create**.

   The Script Wizard - <script name> - Define Answer Choice for <Answer Choice> window appears. Hereinafter, this is referred to as the Define Answer choice window.
3. In the Answer Value field, type the value that you want to store in the database when the user selects this answer choice at runtime.

4. In the Answer Label field, type the value that you want the script end user to see as the answer choice for this question at runtime.

   Oracle Corporation recommends avoiding the use of special characters in this field. Limit answer choice names to alphanumeric characters, spaces, underscores and dashes.

5. If the question type is a checkbox group or multi-select list:

   a. If you want this answer choice to be selected by default for this question at runtime, then in the **Default answer** area, select **Yes**.

   b. If you do not want this answer choice to be selected by default at runtime, select **No**.

   c. Skip to step 12

6. If the question type is a radio button or a drop-down list, perform steps 7 through 11 below.

7. If you want this answer choice to be selected by default for this question at runtime, then in the **Should this be the default answer for this question?** area, select **Yes**.

8. If you do not want this answer choice to be selected by default at runtime, then in the **Should this be the default answer for this question?** area, select **No**.

9. Specify whether distinct branching should be used as a result of the end user selecting this answer choice at runtime, based on the following information, and then click **Next**:

   a. If you want the script to flow to the next panel (as determined by panel sequence listed in the Panel Manager window), select **Go to next default panel**. After clicking **Next**, skip to step 12 in this procedure.

   b. If you want the script to progress to a specific panel when the script user selects this answer choice at runtime, select **Go to a specific panel (ADVANCED)**. After clicking **Next**, you must specify the destination panel.

      The Script Wizard - <script name> - Set Destination Panel for Answer Choice <answer choice name> window appears.

      Hereinafter, this window is referred to as the Set Destination Panel for Answer Choice window.

   c. If you want the script to terminate after exiting the current panel when the script user selects this answer choice at runtime, select **End Script**. After clicking **Next**, skip to step 12 in this procedure.

10. If the panel that you want the script to branch to when this answer choice is selected does not yet exist, then in the Set Destination Panel for Answer Choice window, perform the following:

    a. In the Go to a new or existing panel? area, select **New Placeholder Panel**.

       The New Placeholder Panel field is now enabled.

    b. In the New Placeholder Panel field, type the name you want the destination panel to be called.

       Note: If you are uncertain what the panel name should be, you can enter a panel name that will meaningfully convey the function or business case. Like any other panel, you can always edit this panel name at a later date from the Panel Manager window.

    c. Click **Next**.

       The Answer Manager window appears. The answer choice you defined is included in the list. The Branches to... column for the new answer choice indicates the branching destination.

11. If the panel that you want the script to branch to when this answer choice is selected already exists, then in the Set Destination Panel for Answer Choice window, perform the following:

    a. In the Go to a new or existing panel? area, select **Existing Panel**.

       The Select Existing Panel list is now enabled.

    b. From the Select Existing Panel list, select the destination panel.

    c. Click **Next**.

       The Answer Manager window appears. The answer choice you defined is included in the list. The Branches to... column for the new answer choice indicates the branching destination.

12. When satisfied with the information provided for the first answer choice, then repeat this procedure once more for each answer choice you want to define.

13. Save your work.

Guidelines

- Each answer choice in a question may contain separate distinct branching instructions to control flow in the script.
- 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.

See Also

- Accessing the Answer Manager
- Reordering Answer Choice Sequence
- Editing Existing Answer Choices
- Copying Existing Answer Choices
- Deleting Existing Answer Choices

3.7.4 Editing Existing Answer Choices

Once defined, you can modify an existing answer choice by selecting it from the Answer Manager window.

Use this procedure to edit an existing answer choice for a question.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.
- One or more questions must be defined for the specified panel.
- One or more answer choices must be defined for the specified question.

Steps

1. Access the Answer Manager window for a specific question in a wizard script.
2. In the Answer Manager window, select the record for the appropriate answer choice and click **Edit**.

The Define Answer Choice window appears.

3. To modify the properties for an answer choice, make the appropriate changes and click **Next**.

   Answer choice properties include the answer value, the answer label, the boolean default answer choice option, and instructions for branching based on selection of this answer choice at runtime. For more information, see Creating Answer Choices with the Answer Manager.

   - If using distinct branching, the Set Destination Panel for Answer Choice window appears. Set the destination as appropriate and click **Next**.

     Note: If you modify the branching properties associated with an answer choice, the flow of the script at runtime is modified accordingly.

   The Answer Manager window appears.

4. Save your work.

See Also

- Accessing the Answer Manager
- Reordering Answer Choice Sequence
- Creating Answer Choices with the Answer Manager
- Copying Existing Answer Choices
- Deleting Existing Answer Choices

3.7.5 Copying Existing Answer Choices

You can copy any existing answer choice defined for a question from the Answer Manager window. The copied answer choice contains the exact same characteristics as the original answer choice (including answer choice-specific branching attributes), except that the new answer choice value and label fields are prepended with the words **Copy of** to ensure uniqueness among answer choices for that question. After copying an answer choice, you can modify its properties as appropriate.

Use this procedure to copy an existing answer choice for a question.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.
- One or more questions must be defined for the specified panel.
- One or more answer choices must be defined for the specified question.

Steps

1. Access the Answer Manager window for a specific question in a wizard script.
2. In the Answer Manager window, select the record for the appropriate answer choice and click **Copy**.

   The Define Answer Choice window appears. The properties of the original answer choice appear identically, with the words **Copy of** prepended to the original value in the Answer Name and Answer Value fields.
3. To modify the properties for an answer choice, make the appropriate changes and click **Next**.

   Answer choice properties include the answer value, the answer label, the boolean default answer choice option, and instructions for branching based on selection of this answer choice at runtime. For more information, see Creating Answer Choices with the Answer Manager.

   - If using distinct branching, the Set Destination Panel for Answer Choice window appears. Set the destination as appropriate and click **Next**.

     Note: If you modify the branching properties associated with an answer choice, the flow of the script at runtime is modified accordingly.

   The Answer Manager window appears.
4. Save your work.

See Also

- Accessing the Answer Manager
- Reordering Answer Choice Sequence
- Creating Answer Choices with the Answer Manager

- Editing Existing Answer Choices
- Deleting Existing Answer Choices

3.7.6 Deleting Existing Answer Choices

You can delete any existing answer choices for a question in a wizard script panel from the Answer Manager window.

Note: 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.

Use this procedure to delete an existing answer choice for a question.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.
- One or more questions must be defined for the specified panel.
- One or more answer choices must be defined for the specified question.

Steps

1. Access the Answer Manager window for a specific question in a wizard script.
2. In the Answer Manager window, select the record for the appropriate answer choice and click **Delete**.

   The Confirm Delete window appears, directing you to confirm that you want to delete the selected answer choice from this question.
3. If you want to cancel the delete operation, then from the Confirm Delete window, click **No**.

The Answer Manager window appears. The answer choice remains in the list of available answer choices for this question.

4. If you want to delete the answer choice from this question, then from the Confirm Delete window, click **Yes**.

   The Answer Manager window appears. The answer choice is removed from the question and does not appear in the list of answer choices.

5. Save your work.

See Also

- Accessing the Answer Manager
- Reordering Answer Choice Sequence
- Creating Answer Choices with the Answer Manager
- Editing Existing Answer Choices
- Copying Existing Answer Choices

3.8 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 window).
- 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.
- You can end the script. This uses a default branch to a termination node on the root graph.

Each panel must have an exit panel sequence defined. Thus, one of the preceding three options is defined for each panel in a wizard script.

At the answer choice level, you have the same three choices. 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.

It is important to understand the way the Script Wizard determines script flow to build a script that behaves precisely as your business requirements dictate.

This section includes the following topics:

- Selecting the Next Sequential Panel
- Designating a Specific Panel Destination
- Designating the End of Script Flow
- Understanding Wizard Script Flow Strategies

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Panel Manager
- Using the Question Manager
- Using the Answer Manager
- Saving and Deploying Wizard Scripts

3.8.1 Selecting the Next Sequential Panel

Script routing instructions are relative to the panel sequence listed in the panel manager. Whether selecting the next panel from a panel or from a specific answer choice within a panel, the result will be to flow the script to the next sequential panel.

Thus, assuming your panels are numbered in the sequence in which they are created, if you are selecting this option from Panel 4, you instruct the Scripting Engine to route the script to Panel 5 at runtime.

In real-world scenarios you may need to create additional panels, delete existing panels, or change the sequence of panels initially created using the Script Wizard.

You can select the next sequential panel before a second or subsequent panel has been created.

Use this procedure to select the next sequential panel.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. To select the next sequential panel from a panel:
   a. Access the panel manager for the appropriate script.
   b. Select **Go to the next panel in sequence**.
   c. Save your work.
2. To select the next sequential panel from an answer choice:
   a. Access the Answer Manager window for a specific question in a wizard script.
   b. Select **Go to next default panel**.
   c. Save your work.

See Also

- Designating a Specific Panel Destination
- Designating the End of Script Flow
- Understanding Wizard Script Flow Strategies

3.8.2 Designating a Specific Panel Destination

From a panel or answer choice, you can designate a specific panel to be the next destination in the flow of the script at runtime.

You can designate an already existing panel, or designate a placeholder panel to be the destination. If you designate a placeholder, you must 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**).

Use this procedure to designate a specific panel as the destination at runtime.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. If you want to designate a specific panel destination from a panel, access the panel manager for the appropriate script.
2. If you want to designate a specific panel destination from an answer choice, access the Answer Manager window for a specific question in a wizard script.
3. Select **Go to a specific panel (ADVANCED)**.
4. If the panel that you want the script to branch to does not yet exist, then in the Set Destination Panel window for the appropriate panel or answer choice, perform the following:
   a. In the Go to a new or existing panel? area, select **New Placeholder Panel**.

      The New Placeholder Panel field is now enabled.

   b. In the New Placeholder Panel field, type the name you want the destination panel to be called.

      Note: If you are uncertain what the panel name should be, you can enter a panel name that will meaningfully convey the function or business case. Like any other panel, you can always edit this panel name at a later date from the Panel Manager window.

   c. Click **Next**.

      The Answer Manager window appears.

      If designating a specific panel destination from a panel, the answer manager window is empty. You can define answer choices here, if appropriate. For more information, see Using the Answer Manager.

If designating a specific panel from an answer choice, the answer choice you defined is included in the list, and the Branches to... column for the new answer choice indicates the branching destination.

5. If the panel that you want the script to branch to already exists, then in the Set Destination Panel window for the appropriate panel or answer choice, perform the following:
   a. In the Go to a new or existing panel? area, select **Existing Panel**.

      The Select Existing Panel list is now enabled.

   b. From the Select Existing Panel list, select the destination panel.
   c. Click **Next**.

      The Answer Manager window appears. The answer choice you defined is included in the list. The Branches to... column for the new answer choice indicates the branching destination.
6. Save your work.

See Also

- Selecting the Next Sequential Panel
- Designating the End of Script Flow
- Understanding Wizard Script Flow Strategies

3.8.3 Designating the End of Script Flow

Whether from a panel or from an answer choice within a panel, you can instruct the script to terminate after exiting the current panel. In the wizard script, this causes a default branch to lead from the current panel to a termination node on the root graph.

Use this procedure to designate the end of script flow.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- One or more panels must be defined in the wizard script.

Steps

1. To designate the end of script flow from a panel:

a. Access the panel manager for the appropriate script.

b. Select **End Script.**

c. Save your work.

2. To designate the end of script flow from an answer choice:

a. Access the Answer Manager window for a specific question in a wizard script.

b. Select **End Script.**

c. Save your work.

See Also

- Selecting the Next Sequential Panel
- Designating a Specific Panel Destination
- Designating the End of Script Flow
- Understanding Wizard Script Flow Strategies

3.8.4 Understanding Wizard Script Flow Strategies

This section includes the following topics:

- Distinct Branching and Wizard Scripts
- Placeholder Panels

3.8.4.1 Distinct Branching and Wizard Scripts

When using distinct branching in wizard scripts, you can use one of two 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.

   A third strategy you can employ is to 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, if you create a panel called "ice cream" and create a question called "flavor" with three answer choices, 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 (destinationForChocolate and destinationForVanilla), and for the third flavor answer choice ("other") you leave the default "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 destinationForChocolate panel.

3.8.4.2 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 placeholder panel creation, if you are uncertain what the panel name should be, you can enter a panel name that will meaningfully convey the function or business case for that panel. Like any other panel, you can always edit the placeholder panel name at a later date from the Panel Manager window. As discussed below, you must generally modify the panel's properties in any case.

At the time of creation, placeholders 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

Panel Attribute	Description
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 window 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 window 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 window, 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 window.

See Also

- Selecting the Next Sequential Panel
- Designating a Specific Panel Destination
- Designating the End of Script Flow

3.9 Saving and Deploying Wizard Scripts

You can save a wizard script from any window 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.

Once a wizard script is saved, you can open it from the Script Wizard, edit it, copy it, delete it, convert it to a graphical script, or deploy it.

Scripts cannot be executed until they are deployed from Script Author. From the Script Wizard, scripts are deployed using the **Save, Deploy, and Exit** command.

This section includes the following topics:

See Also

- Getting Started with the Script Wizard
- Creating a Wizard Script
- Opening a Wizard Script from Script Author
- Using the Script Manager
- Using the Panel Manager
- Using the Question Manager
- Using the Answer Manager
- Determining Flow with the Script Wizard

3.9.1 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 four save options.

You can save a wizard script from any window in which the Save button is enabled. The Save options for a wizard script include:

- Renaming a Wizard Script When Saving - this option is available for any script you save.
- Save and Continue Editing
- Save and Exit
- Save, Deploy and Exit

The option to save the current wizard script with a different name is available every time you save a script.

Upon selecting **Save**, if the script contains placeholder panels, the Placeholder Panels Have Not Been Edited window appears. This window 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, simply click **Next**.

Upon selecting **Save** (or upon leaving the placeholder panel warning window, if appropriate), the Save Script <script name> Window appears.

Selecting **Yes** in this window results in the Save Script <script name> window, in which the words **Copy of** are prepended to the existing script name.

The ability to graph a copy of the wizard script is available to the **Save and Exit** and the **Save, Deploy and Exit** save options.

See Also

- Save and Continue Editing
- Save and Exit
- Save, Deploy and Exit

3.9.2 Save and Continue Editing

Selecting **Save and Continue Editing** from the Continue With Script Save window 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 window, from which you can make additional changes.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The wizard script must be syntactically correct and contain at least one panel.

Steps

1. From any point in the Script Wizard that the Save option is enabled, click **Save**.

The Save Script <script name> Window appears.

2. If you want to save a copy of this script with a different name, select **Yes** and click **Next**.
   - The Save Script <script name> window appears. In the Script Name field, the original name of the script appears, prepended by the words **Copy of**.
   - Make any changes and click **Next**.

   The Continue With Script Save window appears.
3. In the Which action do you want to take area, select **Save and continue editing** and click **Next**.

   The Save and Continue Editing Script window appears, confirming that the script changes have been saved.
4. Click **Next**.

   The Panel Manager window appears.
5. In the Which action do you want to take area, select **Save and exit** and click **Next**.
6. In the Which action do you want to take area, select **Save, deploy and exit** and click **Next**.

See Also

- Understanding Save Options
- Save and Exit
- Save, Deploy and Exit

3.9.3 Save and Exit

Selecting **Save and Exit** from the Continue With Script Save window 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 window, 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 window.
  - If you did not graph the script, displays Script Author application with no script open.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The wizard script must be syntactically correct and contain at least one panel.

Steps

1. From any point in the Script Wizard that the Save option is enabled, click **Save**.

   The Save Script <script name> Window appears.

2. If you want to save a copy of this script with a different name, select **Yes** and click **Next**.
   - The Save Script <script name> window appears. In the Script Name field, the original name of the script appears, prepended by the words **Copy of**.
   - Make any changes and click **Next**.

   The Continue With Script Save window appears.

3. In the Which action do you want to take area, select **Save and exit** and click **Next**.

   The Create Graphical Script Copy window appears.

4. If you want to graph the script and immediately view it, then in the Create Graphical Script Copy window, select **Yes** and click **Next**.

   The Script is Complete for <script name> window appears.

5. Click Finish.

   The Script Wizard closes, and the Script Author Java applet appears.

   If you indicated that you wanted to create a graphical copy of the script, the graphical copy of the wizard script is displayed in the Script Author visual layout. The script view shows the entire graph in the window.

If you indicated that you did not want to create a graphical copy of the script, the Script Author frame is visible, but no graphical script is open.

See Also

- Understanding Save Options
- Save and Continue Editing
- Save, Deploy and Exit

3.9.4 Save, Deploy and Exit

Selecting **Save, Deploy and Exit** from the Continue With Script Save window 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 window, 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 window.
  - 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.

Prerequisites

- Script Author must be open. See Accessing the Script Wizard.
- The wizard script must be syntactically correct and contain at least one panel.

Steps

1. From any point in the Script Wizard that the Save option is enabled, click **Save**.

The Save Script <script name> Window appears.

2. If you want to save a copy of this script with a different name, select **Yes** and click **Next**.
   - The Save Script <script name> window appears. In the Script Name field, the original name of the script appears, prepended by the words **Copy of**.
   - Make any changes and click **Next**.

   The Continue With Script Save window appears.

3. In the Which action do you want to take area, select **Save, deploy and exit** and click **Next**.

   The Create Graphical Script Copy window appears.

4. If you want to graph the script and immediately view it, then in the Create Graphical Script Copy window, select **Yes** and click **Next**. If you do not want to create a graphical copy, select **No** and click **Next**.

   The Script is Complete for <script name> window appears.

5. Click Finish.

   The Script Wizard closes, and the Script Author Java applet appears.

   If you indicated that you wanted to create a graphical copy of the script, the graphical copy of the wizard script is displayed in the Script Author visual layout. The script view shows the entire graph in the window.

   If you indicated that you did not want to create a graphical copy of the script, the Script Author frame is visible, but no graphical script is open.

- Script Author attempts to deploy the script to the applications database. The deployment success or failure message appears in the Script Author status area.

See Also

- Understanding Save Options
- Save and Continue Editing
- Save and Exit

4

Using Script Author

This section contains task-based procedures you can perform using Script Author to create and modify graphical and wizard scripts.

The information provided in this section is extensive. Necessarily, some tasks fall into several categories; in these cases, a task may be detailed once in the document below, but referenced under each primary topic in which it is relevant.

This section includes the following topics:

- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.1 Getting Started with Script Author

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
- Obtaining Script Requirements Before Creating a Script
- Determining an Appropriate Script Creation Method

References

- Working with Graphical Script Files
- Defining Global Script Attributes

See Also

- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels

- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.1.1 Recommended Script Creation Flow

To develop a new script you must obtain the requirements for the script, determine your script creation method (graphical tools or Script Wizard), start the script, define global script attributes based on your requirements, use the script creation tool to define the scripting objects needed to implement your business rules, associate properties for each script object, ensure the script flow is defined appropriately, save the script, and deploy it to the applications database. The specific major tasks associated with this process are indicated below.

Steps

1. Obtain the requirements for a script.
2. Determine your script creation method.
3. Start a new script project.
4. Define global script attributes, such as:
   a. For wizard and for graphical scripts, define script properties such as the Script Name, Comments, Script Language
   b. For graphical scripts only, also define boolean script properties, including Footprinting, Answer Collection and the Suspendable property

   c. Global script pre- and post-actions (for graphical scripts)
   d. Script information area (for graphical scripts)
   e. Shortcut buttons (for graphical scripts)
   f. Script Disconnect button (for graphical scripts)
   g. Script Suspend button (for graphical scripts)

5. Define the scripting objects needed to implement your business rules. This includes:
   a. For graphical scripts, inserting appropriate script objects.
   b. For wizard scripts, adding panels.
6. Define properties for each configurable script object. This includes:
   a. For graphical scripts, defining panels, groups, and blocks.
   b. For wizard scripts, defining questions, answer choices, and any required response validation.
7. Ensure script flow is defined appropriately. This includes:
   a. For graphical scripts, insert appropriate branching to meet flow objectives.
   b. 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:
   a. For graphical scripts, saving the script to the file system, or publishing a script to the applications database.
   b. For wizard scripts, saving the script with a different name, if applicable.
9. Deploy the script to the Oracle Applications database:
   a. For graphical scripts, this may include checking the syntax of the script.
   b. 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 earlier 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.

See Also

- Obtaining Script Requirements Before Creating a Script
- Determining an Appropriate Script Creation Method

4.1.2 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.

For more information, see Planning Oracle Scripting Projects.

References

- Getting Started with Script Author
- Defining Global Script Attributes
- Working with Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Controlling Script Flow
- Defining Panel Question UI Controls
- Working with Questions

- Defining Actions
- Defining Commands
- Defining Shortcuts
- Deploying the Script

See Also

- Recommended Script Creation Flow
- Determining an Appropriate Script Creation Method

4.1.3 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 re-importing 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.

See Also

- Recommended Script Creation Flow
- Obtaining Script Requirements Before Creating a Script

4.2 Working with Graphical Script Files

You can perform the following tasks:

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode

- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

References

- Obtaining Script Requirements Before Creating a Script
- Defining Global Script Attributes

See Also

4.2.1 Starting a New Script

Use this procedure to create a new graphical script.

Prerequisites

None

Steps

1. Choose **File** > **New**, or click the **New Script** icon in the toolbar.

   Note: The **New Script** icon is the first icon from the left of the toolbar, located immediately below the menus in the Script Author interface.

   The New Script dialog window appears.

2. In the New Script dialog window, click **Graphical Script**.

   The New Script dialog window closes.

   A new, untitled graphical script opens. The start node appears on the canvas.

References

- Defining Global Script Attributes

See Also

- Opening an Existing Script

- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.2 Opening an Existing Script

All scripts created for Oracle Scripting are opened from Script Author.

- From within Script Author, graphical scripts can be opened from the file system on any local volume. Published or deployed scripts can be opened from the applications database for your environment.
- Wizard scripts are stored as published and deployed scripts only in the applications database for your environment, and can only be opened from within the Script Wizard component of Script Author. Wizard scripts cannot be saved on your local file system or as a separate file on a network, and therefore cannot be opened from a file stored on the file system or network.

Within the database, there is a distinction between *published* scripts and *deployed* scripts. Published scripts are listed in IES_PUBLISHED_SCRIPTS table within the Oracle Applications schema. This is essentially equivalent to storing a script on your local filing system, except it can also be accessed by any user with the Scripting Administrator responsibility for the designated environment.

Deployed scripts are listed in IES_DEPLOYED_SCRIPTS. The only way to store a script here is to explicitly deploy it from Script Author, after which the script is executable using the Scripting Engine. (Additional survey campaign administration steps are required to launch any script in a Web browser using the Scripting Engine Web interface.)

Use these procedures to open a saved script from the file system or from the applications database.

This section includes the following topics:

References

- Defining Global Script Attributes
- For information regarding opening wizard scripts, see Using the Script Wizard > Opening a Wizard Script from Script Author

See Also

- Starting a New Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.2.1 Opening an Existing Script from the File System

Scripts can be opened from the file system on any local volume, or from the applications database for your environment.

Use this procedure to open a saved script from the file system.

Prerequisites

None

Steps

1. Choose **File** > **Open** or click the **Open a Script** icon in the toolbar.

> **Note:** The **Open a Script** icon looks like a file folder with a blue arrow, showing the folder being opened.

The Open Script dialog window appears.

2. In the Open Script dialog window, click **Graphical Script**.

   The ScriptChooser window for Script Author appears.

3. Click the **File System** tab from the ScriptChooser window, if not already selected.

4. Optionally, from the **Location** list, select the appropriate local volume.

5. If the script file you want to open does not contain a file extension of .SCRIPT or .SCR, then from the File Type list, select **All Files (*.*)**.

6. From the **Files** list, locate and select the script that you want to open, and then click **Open**.

   The ScriptChooser window closes.

   The selected script appears on the canvas.

> **Note:** If you wish to preserve the original script, ensure you change the name of the just-opened script in both the global script properties (**File > Script Properties**) and in the file system (**File > Save As**).

References

- Defining Global Script Attributes

See Also

- Opening an Existing Script from the Applications Database

4.2.2.2 Opening an Existing Script from the Applications Database

Scripts can be opened from the file system on any local volume, or from the applications database for your environment.

Use this procedure to open a published or deployed script from the applications database.

Prerequisites

A script must already exist in the applications database, either as a published script or as a deployed script.

Steps

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

   The **Open Script** dialog window appears.

2. In the Open Script dialog window, click **Graphical Script**.

   The ScriptChooser window for Script Author appears.

3. Click the **Database** tab from the **Script Chooser** window, if not already selected.

4. If you want to open a script that is published, select **Published**.

   If you want to open a script that is deployed to the applications database (for example, a production script), select **Deployed**.

5. Click **Refresh List** to ensure the displayed list of published or deployed scripts is updated.

   The Scripts list is updated.

6. From the Scripts list, locate the appropriate script (uniquely identified by the combination of script name and script language), and then click **Open**.

   The ScriptChooser window closes.

   If the script was published or deployed from an earlier version of Script Author, the 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.

> **Note:** If you wish to preserve the original script in the database, ensure you change the name of the just-opened script in the global script properties (**File > Script Properties**). If you will be saving the script to your local file system, change the file name of the script as well (**File > Save As**).

References

- Defining Global Script Attributes

See Also

- Opening an Existing Script from the File System

4.2.3 Saving a Script to the File System or Database

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 (whether complete, or in progress) in the database (rather than in a file system), ensuring that it is available to view or modify in the future. A published script is not deployed (it is not executable). You must explicitly deploy a script from Script Author before you can execute it.

When you take action to save a script (in the ScriptChooser window of the graphical Script Author user interface), you must designate a location to which you want the script to be saved, and determine whether to save it to a file system or publish it to the database.

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.

This section includes the following topics:

- Saving a Script to the File System
- Publishing a Script to the Applications Database

References

- Defining Global Script Attributes
- Reversing All Changes Since the Last Save Command

See Also

- Starting a New Script
- Opening an Existing Script from the File System
- Opening an Existing Script from the Applications Database
- Reversing All Changes Since the Last Save Command

- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.3.1 Saving a Script to the File System

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.

The name which you provide when you save the script in the *file system* is distinct from the name by which the script is identified in the *script properties*. It is the name in the script properties which identifies the script in the database (and in the Script Chooser window of the agent interface at runtime).

In contrast, the file name is used strictly to reference the script in the file system. You may wish to use the same name in the script properties that you identify in this step for the file name.

Use this procedure to save a graphical script to the file system of a local volume or mounted storage medium.

Prerequisites

None

Steps

1. To save a new, untitled script, perform the following steps:
   a. Select **File** > **Save** or click the **Save Script** icon in the toolbar.

      The ScriptChooser window appears.

   b. To save the script in the file system, select the **File System** tab.

   c. Using the **Location** and **Files** fields, navigate to the local or network location where you want to save the script.

d. In the **File Type** field, select **Oracle CRM script file (.script, .scr)**.

e. 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.

f. Click **Save**.

   The ScriptChooser window closes. The script is saved to the filing system in the location you designated.

2. To save a previously existing script using its current name and location, select **File** > **Save,** or click the **Save Script** icon in the toolbar.

3. To save a previously existing script using a different name or location, perform the following steps:

   a. Select **File** > **Save As**.

      The ScriptChooser window appears.

   b. Select the **File System** tab.

   c. Using the **Location** and **Files** fields, navigate to the local or network location where you want to save the script.

   d. In the **File Type** field, select **Oracle CRM script file (.script, .scr)**.

   e. 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.

   f. Click **Save**.

      The ScriptChooser window closes. The script is saved to the filing system in the location you designated.

Guidelines

- 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).

References

- Defining Global Script Attributes

See Also

- Publishing a Script to the Applications Database

4.2.3.2 Publishing a Script to the Applications Database

Published scripts are stored in the IES_DEV_SCRIPTS table of the applications database, but are not executable. This method of storing scripts serves the same purpose as storing to the file system. Thus, scripts in progress or completed scripts not ready for production can be published to the applications 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 (short for American English). 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.

Use this procedure to save a graphical script to the applications database.

Prerequisites

- Defining the Script Name
- Defining Script Language

Steps

1. To publish a new, untitled script, perform the following steps:
   a. Select **File** > **Save** or click the **Save Script** icon in the toolbar.

      The ScriptChooser window appears.

   b. To publish the script to the applications database, select the **Database** tab.

      The global script name and language properties automatically populate in the Script Name and language fields. These cannot be changed.

   c. Click **Save**.

      The ScriptChooser window closes. The script is published to the applications database, and can be opened, modified, or deployed at will.
2. To publish a previously existing script, perform the following steps:
   a. Select **File** > **Save As**.

      The ScriptChooser window appears.

   b. To publish the script to the applications database, select the **Database** tab.

      The global script name and language properties automatically populate in the Script Name and language fields. These cannot be changed.

   c. Click **Save**.

      The ScriptChooser window closes. The script is published to the applications database, and can be opened, modified, or deployed at will.

References

- Defining Global Script Attributes
- Defining the Script Name

See Also

- Saving a Script to the File System

4.2.4 Reversing All Changes Since the Last Save Command

When working with graphical scripts, you can revert to the last version you saved in the file system, or to the last backup version saved. A backup version is made to each graphical script automatically when you open it in a newer Script Author version.

The command to reverse changes since the last save or backup is unavailable for scripts saved to the database, since this feature relies on opening the saved or backup versions of the script on the file system, respectively. Scripts saved in the database (published or deployed scripts) do not retain saved or backup script versions.

Use this procedure to undo all changes made to a graphical script since you last saved the script to the file system.

Prerequisites

A version of the script must be saved or backed up to the file system.

Steps

1. If you want to revert to the last version of the graphical script that was explicitly saved, choose **File** > **Revert To > Last Save**.
2. If you want to revert to the last backup version of the graphical script, choose **File** > **Revert To > Last Backup**.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session

- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.5 Importing a Script

When you import a script, the entire script (the root graph, and all sub-graphs) represented by that script is placed into your current script, within a group. By default, the imported group is named using the global properties of the imported script.

Use this procedure to import a previously existing script into the current script.

> **Note:** Every Script Author script is customized. It is the script developer's responsibility to ensure **imported** scripts will function in the target environment. 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. *Do not assume imported scripts will function without thorough unit testing.*

Prerequisites

None

Steps

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

   The Open window appears.

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, if the script you want to import has no file extension, or has any extension other than .SCRIPT, select **All Files (*.*)**.

   > **Note:** Script Author scripts with a file extension other than .SCRIPT *will not appear in the Open window* unless you first change the File Type option to **All Files (*.*)**.

4. In the Files window, select the script file name and then click **Open**.

   The imported script appears on the canvas as a group.

Guidelines

The import command imports *all* objects in an existing script. If you wish to import only a portion of a script, you import the entire script as described earlier, and discard the unwanted portions, or you can export only the required portions from an existing script, and then import it as described earlier.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.6 Exporting a Script Group as a Separate Script File

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.

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

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. 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. *Do not assume exported scripts will function without thorough unit testing.*

Prerequisites

None

Steps

1. On the canvas, select a group.
2. Choose **File > Export...**.

   The Save window appears.
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.
6. Continue your work within the script or save your work and exit the script.

Guidelines

- The file name you provide to the exported script is different than 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.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.7 Printing a Graph on the Script Author Canvas

Use this procedure to print a graph displayed on the Script Author canvas.

Prerequisites

None

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.

References

- Fitting a Script Layout in the Canvas
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block
- Aligning Objects

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.8 Closing a Script

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

Prerequisites

None

Steps

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

   A prompt appears in the **Select an Option** window asking if you wish to save the changes in the graph.

2. Select the appropriate choice from the prompt.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.9 Toggling Sticky Mode

In every Script Author session, the *sticky mode* feature is enabled by default. In sticky mode, when a script object is selected 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 on the canvas, another instance of the "stuck" object type will appear. For example, if you select the panel object in the toolbar, then each subsequent time you click on the canvas, a new panel object is inserted. You cannot reposition existing objects on the canvas, place a different object type, or connect objects with branches until you explicitly make a subsequent selection from the toolbar.

When sticky mode is enabled, if you select a different object type or branch, then that option remains as the default selection until you explicitly select another tool from the toolbar. When sticky mode is enabled, the only way to reposition existing

objects on the canvas is to enable Toggle Select mode from the toolbar. This is also recommended when you want to access an object's properties in sticky mode.

If you prefer to assign attributes to each object as you create it, leaving sticky mode on can result in the unwanted effect of inserting multiple objects when your intended result may be to double-click a created object to associate properties to it.

You can disable or enable sticky mode at any time in a Script Author session, at your discretion.

Use this procedure to toggle sticky mode on or off.

Prerequisites

None

Steps

1. Select the **File** menu by clicking **File** or pressing the keys **ALT-F**.

   The File menu expands.

2. View the **Sticky Mode** setting.

   If sticky mode is enabled, this option is selected. If sticky mode is disabled, this option is cleared.

3. To enable sticky mode, select the **Sticky Mode** option.

   Alternatively, to toggle the current setting, press **ALT-M**.

   The **File** menu closes, and sticky mode is enabled. To verify, select the **File** menu and ensure this option is selected.

4. To disable sticky mode, when the **Sticky Mode** option is selected, click to clear this option.

   Alternatively, to toggle the current setting, press **ALT-M**.

   The **File** menu closes, and sticky mode is toggled off. To verify, select the **File** menu and ensure this option is cleared.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database

- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Exiting Script Author
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.10 Exiting Script Author

Use this procedure to exit Script Author. If you have unsaved changes, you will be asked whether you want to save the changes.

Prerequisites

None

Steps

1. From the **File** menu, choose **Exit**, or press **Ctrl-Q.**

   If you have unsaved changes, a prompt appears in the **Select an Option** window asking if you wish to save the changes in the graph.

2. Select the appropriate choice from the prompt.

References

Saving a Script to the File System

See Also

- Starting a New Script
- Defining Global Script Attributes
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command

- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Recovering from an Expired Session
- Packaging Java Bean or Custom Java Code Into a JAR File

4.2.11 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 Corporation 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.

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.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File

4.2.12 Packaging Java Bean or Custom Java Code Into a JAR File

Oracle Scripting 11*i* allows the substitution of a custom Java Bean user interface in place of a standard Scripting panel.

Oracle Scripting 11*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 Corporation 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 or with the Caching Architecture, 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
   ```

6. At the command line, start the JAR utility. For example:

   ```
   C:\jdk1.1.8\bin>JAR.EXE).
   ```

7. 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.

See Also

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session

4.3 Working with Script Author Toolbars

When working with graphical scripts, the Script Author UI includes two toolbars immediately above the canvas. The Object and Branch toolbar allows you to insert objects into the script, or branches connecting objects. In most cases, you must then associate properties with the objects or branches.

The Navigation and Alignment toolbar helps you navigate between different graphs within the script, control the view or magnification of the portion of the script currently visible on the canvas, or align objects placed on the canvas.

You can perform the following tasks:

- Inserting an Object
- Inserting a Branch
- Selecting Objects
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block
- Fitting a Script Layout in the Canvas
- Aligning Objects

References

- Selecting Objects
- Deselecting Objects
- Moving Objects
- Changing the Object Connected to a Branch
- Deleting Panels, Groups, Blocks and Termination Nodes
- Deleting Branches

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.3.1 Inserting an Object

In a graphical script, Script Author uses three configurable objects (panels, blocks and groups) and two nonconfigurable objects (the start and termination nodes). These objects must be connected with branches to direct the flow of the script at runtime.

- If you want to display information (such as text, an image, a hyperlink, a variable, or question UI controls) to the script user at runtime, or provide a method to accept information (answers) from the script end user, then insert a panel.
- If you want to query, update, or insert information in database tables, or insert a command or API with a clear visual indicator, then insert a block.
- If you want to logically group a section of the script's functionality, access a section of the script in runtime using a shortcut button, or group functionality that is the target of a Java method in runtime associated with an indeterminate branch, then insert a group.
- If you want to direct the script flow to the end of processing on that graph, then insert a termination node.

You can perform the following tasks:

- Setting Properties Window to Pop Up at Object Creation
- Inserting a Panel
- Inserting a Block
- Inserting a Group
- Inserting a Termination Node

References

- Inserting a Branch
- Setting Properties Window to Pop Up at Branch Creation

See Also

- Inserting a Branch
- Selecting Objects
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block

- Fitting a Script Layout in the Canvas
- Aligning Objects

4.3.1.1 Setting Properties Window to Pop Up at Object Creation

Use this procedure to set up Script Author to automatically display the Properties window when you insert a panel, group, or block onto the canvas. This feature can be particularly helpful when initially inserting panels, as panels require at least one answer be defined. Groups and Blocks also require appropriate termination and branching within their subgraphs. For more information, see Minimum Requirements for Any Graph.

This display property, once set, is retained for the duration of the Script Author session.

Prerequisites

None

Steps

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

See Also

- Inserting a Panel
- Inserting a Block
- Inserting a Group
- Inserting a Termination Node

4.3.1.2 Inserting a Termination Node

Use this procedure to insert a termination node onto the canvas. There are no properties to associate for this object. For more information, see Graphical Script Objects > Termination Nodes.

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

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Termination Point Insertion Mode** tool.

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

2. On the canvas, click where you want to insert the termination node.

See Also

- Setting Properties Window to Pop Up at Object Creation
- Inserting a Panel
- Inserting a Block
- Inserting a Group

4.3.2 Inserting a Branch

Branches control script flow at runtime, based on the branch type used.

In a graphical script, Script Author provides the ability to connect objects on the canvas with any of four branch types: default, distinct, conditional, or indeterminate.

Scripts created using the Script Wizard can include default and distinct branch types. You must graph a wizard script and modify it to include other branch types.

A branch always begins from an existing Script Author object on the canvas. All branches but the Indeterminate branch must also end at another scripting object on the canvas.

- If you want to progress the script to the next object in sequence when only a single branch is required, then insert a default branch.
- If you want to provide a default path for progression of the script for any business case *not* accounted for by other outgoing branches from that script object, then insert a default branch.
- If you want to provide a distinct path if a specific answer is selected at runtime, then insert a distinct branch.

- If you want to provide a path that is only followed when a conditional Java expression is evaluated at runtime and is determined to be true, then insert a conditional branch. This branch type requires you to associate a command with the branch.
- If you want to provide a path that is not determined until a Java command is evaluated at runtime, then insert an indeterminate branch. This branch type requires you to associate a Java command with the branch; the command must return a value for each condition evaluated. Each return value in the method must be a valid destination within the script. For more information, see the topic Using the Indeterminate Branch in *Oracle Scripting Developer's Guide*.

> **Note:** If the destination to which you want to route the user upon a given condition is a sibling object (an object on the same graph), then you can use the object name as the return value of the condition in the Java method. However, if the script object to which you want to route at runtime is not on the same graph, then you must route to a group, using its shortcut property. In this case, the return value must be the name of the shortcut given to the group. If you want to jump to a panel or block, then place that panel or block as the first configurable object in a group, and route to the group's shortcut.

References

- Defining Branches
- Working with Branches on the Canvas
- Defining a Java Command

See Also

- Inserting an Object
- Selecting Objects
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block
- Fitting a Script Layout in the Canvas
- Aligning Objects

4.4 Viewing Objects on the Canvas

You can perform the following tasks:

- Fitting a Script Layout in the Canvas
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block
- Aligning Objects

References

- Working with Objects on the Canvas

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands

- Deploying the Script

4.4.1 Fitting a Script Layout in the Canvas

You can increase or decrease the magnification level of the objects on the canvas using the Zoom tools. These allow you to focus the details of the portion of a script you are editing or reviewing, allowing you to zoom in (increase the magnification), zoom out (decrease the size of all objects on the canvas), or zoom to 100% (reset to the standard size for all objects on the canvas).

Prerequisites

None

Steps

Do one of the following:

- To increase the magnification, click the **Zoom In** tool on Navigation and Alignment toolbar.
- To decrease the magnification, click the **Zoom Out** tool on the Navigation and Alignment toolbar.
- To set the magnification to 100%, setting the view to show the actual size of the objects on the canvas, click the **Zoom to 100%** tool on the Navigation and Alignment toolbar.
- To resize the script layout to the size of the canvas, click the **Zoom to Fit Graph in Window** tool on the Navigation and Alignment toolbar.

See Also

- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block
- Aligning Objects

4.4.2 Navigating to the Root Graph

The root graph is the first graph that appears when a graphical script is opened. Processing for any script at runtime 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. Adding a group or block to the root graph of a script creates a subgraph. Subgraphs

can be nested to whatever degree is required. When viewing the canvas of a subgraph, you may wish to return the view to the root graph.

Use this procedure to navigate from any sub-graph in the canvas to the root graph.

Prerequisites

For this procedure to be effective, you must be viewing a graph other than the root graph.

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. In the Navigation and Alignment toolbar, select the **Go to Root Graph** tool.

The root graph appears in the canvas.

References

- Inserting an Object
- Inserting a Branch
- Selecting Objects

See Also

- Fitting a Script Layout in the Canvas
- Drilling Down Into or Up From a Group or Block
- Aligning Objects

4.4.3 Drilling Down Into or Up From a Group or Block

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

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. To drill down, select a group or block on the canvas and then, in the Navigation and Alignment toolbar, click **Go down into child graph**.

1. To drill up one level, in the Navigation and Alignment toolbar, click **Go up to parent graph**.
2. To go to the top level graph, in the Navigation and Alignment toolbar, click **Go to root graph**.

References

- Inserting an Object
- Inserting a Branch
- Selecting Objects

See Also

- Fitting a Script Layout in the Canvas
- Navigating to the Root Graph
- Aligning Objects

4.4.4 Aligning Objects

Use this procedure to align objects vertically or horizontally.

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select the objects that you want to align.
3. In the Navigation and Alignment toolbar, click **Align Selected Objects Vertically** or **Align Selected Objects Horizontally**.

References

- Inserting an Object
- Inserting a Branch
- Selecting Objects
- Navigating to the Root Graph

See Also

- Fitting a Script Layout in the Canvas
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block

4.5 Working with Objects on the Canvas

You can perform the following tasks:

- Selecting Objects
- Deselecting Objects
- Moving Objects
- Changing the Object Connected to a Branch
- Deleting Panels, Groups, Blocks and Termination Nodes
- Deleting Branches

4.5.1 Selecting Objects

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

Prerequisites

All of the objects to be selected must be on the same canvas.

Steps

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

     All objects in or on the selection border are selected. This includes branches.
   - To add an object to a selection, press the Control key and click on the object you want to add to the selection.

See Also

- Deselecting Objects
- Moving Objects
- Changing the Object Connected to a Branch
- Deleting Panels, Groups, Blocks and Termination Nodes
- Deleting Branches

References

- Inserting an Object
- Inserting a Branch
- Selecting Objects
- Navigating to the Root Graph
- Drilling Down Into or Up From a Group or Block
- Fitting a Script Layout in the Canvas
- Aligning Objects

4.5.2 Deselecting Objects

Use this procedure to deselect one or more selected objects.

Prerequisites

None

Steps

Do one of the following:

- To deselect an object, click outside the object.
- To deselect one of several selected objects, press the Control key and click on the object you want to deselect.
- To deselect all selected objects, click on the canvas away from any objects.

See Also

- Selecting Objects

- Moving Objects
- Changing the Object Connected to a Branch
- Deleting Panels, Groups, Blocks and Termination Nodes
- Deleting Branches

4.5.3 Moving Objects

Use this procedure to move panels, groups, and blocks on the canvas.

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select one or more objects.
3. Drag the object to the desired location.

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

   If a branch crosses over other objects on the canvas, there will be no negative effect at runtime. However, if desired, you can add corners to the branch to make the graph more visually appealing or easy to view the workflow.

References

- Adding a Corner to an Existing Branch
- Working with Branches on the Canvas

See Also

- Selecting Objects
- Deselecting Objects
- Changing the Object Connected to a Branch
- Deleting Panels, Groups, Blocks and Termination Nodes
- Deleting Branches

4.5.4 Changing the Object Connected to a Branch

Use this procedure to change the destination object for a branch. For example, if a branch is connected to a panel called Object A and you want to connect it instead to Object B, perform this task.

Use this procedure to change the destination object for a branch.

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select the destination object for a branch.

> **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.

3. Choose **Edit** > **Delete** to delete the object currently connected to the branch.

   The object that was previously the destination of the branch is deleted.
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.

   The branch is redrawn to the new destination object.

See Also

- Selecting Objects
- Deselecting Objects
- Moving Objects
- Deleting Panels, Groups, Blocks and Termination Nodes
- Deleting Branches

4.5.5 Deleting Panels, Groups, Blocks and Termination Nodes

Use this procedure to delete objects from the canvas. Start nodes cannot be deleted using this procedure. The only way a start node can be deleted is to delete its parent object (group or block). Obviously, the start node of the root graph cannot be deleted.

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

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select one or more objects.
3. Choose **Edit** > **Delete**.

See Also

- Selecting Objects
- Deselecting Objects
- Moving Objects
- Changing the Object Connected to a Branch
- Deleting Branches

4.5.6 Deleting Branches

Use this procedure to delete branches from the canvas.

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

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select a branch.

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

See Also

- Selecting Objects
- Deselecting Objects
- Moving Objects
- Changing the Object Connected to a Branch
- Deleting Panels, Groups, Blocks and Termination Nodes

4.6 Working with Branches on the Canvas

Once a branch has been defined on the canvas, it can be modified visually by adding and moving corners, to accommodate the objects placed on the canvas and create a clear picture of the intended flow of the script.

Typically, corners are added to a branch to clarify the script flow and to avoid intersecting objects or branches on the canvas. This modification is strictly visual, and has no affect on the function of the branch.

Working with branches, you can perform the following tasks:

- Inserting a Straight Branch
- Inserting a Branch with Corners
- Adding a Corner to an Existing Branch
- Moving a Corner of a Branch
- Deleting a Corner from a Branch

References

- Changing the Object Connected to a Branch

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.6.1 Inserting a Straight Branch

Use this procedure to insert a straight branch onto the canvas to connect two objects, with no corners.

Prerequisites

To insert a branch, you must have at least two object on the canvas.

Steps

1. In the Object and Branch toolbar, click the appropriate branch type tool.

When the pointer is over the canvas, the pointer is a crosshair (+).

2. On the canvas, drag the pointer from the source object to the destination object.
   - In the case of an Indeterminate branch, an empty space on the canvas is the destination.
   - When you release the pointer over the destination object, the branch arrow appears. The branch is red. This indicates that the branch is currently selected.
3. Optionally, define the branch name.

Guidelines

- Script Author will not allow you to start a branch unless you choose a source object. You cannot start a branch from a blank area of the canvas.
- Script Author will not allow you to connect a start node with any branch type other than a default branch. (You will get an invalid edge error message.)
- If defining a distinct branch, the source object must be a panel.
- If defining a conditional branch, you must then associate a condition with the branch. This is typically a Script Author Java command that provides a condition which, if true, is the branch of the script taken at runtime.
- If defining an indeterminate branch, you must then associate an expression with the branch. This is typically a Script Author Java command that provides an expression containing one or more conditions. For each condition, a destination in the script is provided as a return value.

See Also

- Inserting a Branch with Corners
- Adding a Corner to an Existing Branch
- Moving a Corner of a Branch
- Deleting a Corner from a Branch

4.6.2 Inserting a Branch with Corners

Use this procedure to insert a branch with corners.

Prerequisites

To insert a branch, you must have at least two object on the canvas.

Steps

1. In the Object and Branch toolbar, click a branch tool.

   When the pointer is over the canvas, the pointer is a crosshair (+).

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

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

4. Click the destination object.

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

Guidelines

Although this task describes creating a single corner, you can include any number of corners required in between the source and destination objects.

See Also

- Inserting a Straight Branch
- Adding a Corner to an Existing Branch
- Moving a Corner of a Branch
- Deleting a Corner from a Branch

4.6.3 Adding a Corner to an Existing Branch

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

Prerequisites

A branch must exist to which you want to add one or more corners. Typically, corners are added to a branch to clarify the script flow and avoid objects or branches intersecting on the canvas. This modification is strictly visual, and has no effect on the function of the branch.

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, click to select a branch.

   When the branch is selected, the branch is red.
3. Drag the branch to the desired location and then release the pointer.

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

See Also

- Inserting a Straight Branch
- Inserting a Branch with Corners
- Moving a Corner of a Branch
- Deleting a Corner from a Branch

4.6.4 Moving a Corner of a Branch

Use this procedure to move a corner of a branch.

Prerequisites

An existing branch in a graphical script must contain at least one corner.

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select a branch.
3. Select a branch corner.

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

See Also

- Inserting a Straight Branch
- Inserting a Branch with Corners
- Adding a Corner to an Existing Branch

- Deleting a Corner from a Branch

4.6.5 Deleting a Corner from a Branch

Use this procedure to delete a corner from a branch.

Prerequisites

An existing branch in a graphical script must contain at least one corner.

Steps

1. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
2. On the canvas, select a branch.
3. Select a branch corner.

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

See Also

- Inserting a Straight Branch
- Inserting a Branch with Corners
- Adding a Corner to an Existing Branch
- Moving a Corner of a Branch

4.7 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.

Global Script Attribute	Runtime Interface	Description
Script pre- and post-actions	All	Script Author commands that execute at the beginning or end of a script.
Script information area	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. Formerly referred to as the Static Panel.
Shortcut buttons	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. The area in which shortcut buttons appear at runtime is now known as the shortcut button area, but is sometimes referred to as the Shortcut Panel.

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.

You can perform the following tasks:

- Designating Global Script Properties
- Defining Global Script Pre- and Post-Actions
- Defining the Script Information Area
- Defining Shortcut Buttons
- Programming the Script Disconnect Button

References

- Starting a New Script
- Opening an Existing Script
- Saving a Script to the File System or Database
- Reversing All Changes Since the Last Save Command
- Importing a Script
- Exporting a Script Group as a Separate Script File
- Printing a Graph on the Script Author Canvas
- Closing a Script
- Toggling Sticky Mode
- Exiting Script Author
- Recovering from an Expired Session

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls

- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.7.1 Designating Global Script Properties

Global script properties control the way scripts behave from a database perspective and in runtime, and affect data that is collected when scripts execute in the Scripting Engine. These script properties are all accessible in Script Author by selecting **File > Script Properties**. The first set of global script attributes are the script properties listed below:

- Defining the Script Name
- Defining Script Comments
- Defining Script Language
- Enabling or Disabling Footprinting
- Enabling or Disabling Answer Collection
- Enabling or Disabling the Suspend Button

See Also

- Defining Global Script Pre- and Post-Actions
- Defining the Script Information Area
- Defining Shortcut Buttons
- Programming the Script Disconnect Button

4.7.1.1 Defining the Script Name

Use this procedure to define the global name of the script. This is the name under which the script is saved and referenced by the database. Do not include the percent character, single quotes, or double quotes in the global script name, as these are special database characters. The name you provide in this step is displayed (along with the designated script language) in the Script Chooser window at runtime.

The global script name is distinct from the script name from a file system perspective. To avoid confusion you may wish to use the same name in the script properties that you identify in the file system. If so, you must also avoid using spaces, slashes, or backslashes in the name, which are not permitted in the file system of some operating systems. Using a .SCRIPT or .SCR file extension in the global script name *is not necessary*, although it will cause no harm.

> **Note:** 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.

Prerequisites

Create or open a script. You must have a script open to designate the script name.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

   The Properties window appears.
2. In the Script tree, select **Properties**.

   The Properties pane appears.
3. In the Properties pane, in the Name field, type the name of the script.

   Do not include the percent character, single quotes, or double quotes in this name.
4. Optionally, add or modify other script property information.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

References

- Starting a New Script

- Opening an Existing Script from the File System
- Opening an Existing Script from the Applications Database

See Also

- Defining Script Comments
- Defining Script Language
- Enabling or Disabling Footprinting
- Enabling or Disabling Answer Collection
- Enabling or Disabling the Suspend Button

4.7.1.2 Defining Script Comments

Use this procedure to include a comment associated with the global script. If script developers routinely open scripts from the database, or if multiple developers work on the same script, providing script comments may be helpful in identifying particular script versions or features. There is no maximum character limit to a comment and no character restrictions.

Prerequisites

Create or open a script. You must have a script open to define script comments.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

   The Properties window appears.
2. In the Script tree, select **Properties**.

   The Properties pane appears.
3. In the Properties pane, in the Comments field, type the desired comment.
4. Optionally, add or modify other script property information.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining the Script Name
- Defining Script Language
- Enabling or Disabling Footprinting
- Enabling or Disabling Answer Collection
- Enabling or Disabling the Suspend Button

4.7.1.3 Defining Script Language

Use this procedure to designate a script language associated with a script. If using American English, this step is not necessary, as the value AMERICAN is the default. The Script Author script language setting can use any language supported by Oracle Applications as designated in the FND_LANGUAGES table.

> **Note:** Changing the script language setting *does not translate GUI elements or script elements.* The script language setting is intended solely to communicate to agents attempting to launch a script in what language that script has been created. If an enterprise has a business requirement to translate an English script to Spanish, for example, a script developer can open the version labeled AMERICAN, change the setting to SPANISH, and then must manually translate all panel text, answer lookup values, panel names or labels, or any customized aspect of a script. The name of the script can be left the same as the English version. When the Spanish version is deployed to the database, both will list in the Script Chooser.

Prerequisites

Create or open a script. You must have a script open to designate the script language.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or

- In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

The Properties window appears.

2. In the Script tree, select **Properties**.

   The Properties pane appears.

3. In the Properties pane, from the Script language list, select the appropriate language to indicate in what language the script is written.

Guidelines

- This list is derived from the FND_LANGUAGES table of the applications database.
- Any uni-directional language used in FND_LANGUAGES is supported.

See Also

- Defining the Script Name
- Defining Script Comments
- Enabling or Disabling Footprinting
- Enabling or Disabling Answer Collection
- Enabling or Disabling the Suspend Button

4.7.1.4 Enabling or Disabling 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.

Use this procedure to set footprinting for the appropriate Oracle Scripting runtime component. This is accomplished on a per-script basis from Script Author.

Prerequisites

- To enable this feature, the Footprinting option must have been previously disabled.
- To disable this feature for a wizard script, you must first graph the script.
- Create or open a graphical script. You must have a script open to set footprinting.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

   The Properties window appears.
2. In the Script tree, select **Properties**.

   The Properties pane appears.
3. In the Properties pane, to record the start time and end time (in milliseconds) for every script panel that is reached during an agent interaction, select **Footprinting**.
4. To disable the footprinting feature, clear the **Footprinting** option.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining the Script Name
- Defining Script Comments
- Defining Script Language
- Enabling or Disabling Answer Collection
- Enabling or Disabling the Suspend Button

4.7.1.5 Enabling or Disabling Answer Collection

Use this procedure to have Oracle Scripting record the responses (answers) to questions in any panel provided at runtime by script end users of any Scripting Engine interface.

Prerequisites

- To enable this feature, the Answer Collection option must have been previously disabled.
- To disable this feature for a wizard script, you must first graph the script.
- Create or open a graphical script. You must have a script open to set answer collection.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

   The Properties window appears.

   In the Script tree, select **Properties**.The Properties pane appears.
2. In the Properties pane, to record the answers to the script panels enabled during an agent interaction, select **Answer Collection**.

   Note: Answer collection will also take place if you select the Footprinting option, even if the Answer Collection option is cleared.
3. To disable the answer collection feature, clear the **Footprinting** option.
4. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining the Script Name
- Defining Script Comments

4.7.1.6 Enabling or Disabling the Suspend Button

By default, the Suspendable global script property is enabled for all scripts (graphical and wizard scripts). When the IES : Display Suspend Button on Script Frame profile option is set to **True**, and the Suspendable property for a script is enabled, a scripting transaction in the agent interface can be suspended, and later resumed.

The Suspend button appears, at runtime, to the left of the Disconnect button in the script frame of the Scripting Engine agent interface. Clicking this button suspends a script transaction, taking a snapshot in time of the state of the script when the button is clicked.

The only way to disable the Suspend button for a wizard script is to graph the script and then follow the procedure below. Note that, if you make no other changes to the wizard script, you can still access it via the Script Wizard to view or modify the script. The only caveat to this approach is that you must graph the script and disable the Suspend button anew each time you change the wizard script.

Use this procedure to enable or disable the Suspend button in the Scripting Engine agent interface.

Prerequisites

- To enable this feature, the Suspendable option must have been previously disabled.
- To disable this feature for a wizard script, you must first graph the script.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

   The Properties window appears.
2. In the Script tree, select **Properties**.

The Properties pane appears.

In the Properties pane, to enable the Suspend button, select **Suspendable**.

3. To disable the Suspend button, clear the Suspendable option.
4. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window for the script.

See Also

- Defining the Script Name
- Defining Script Comments
- Defining Script Language
- Enabling or Disabling Footprinting
- Enabling or Disabling Answer Collection

4.7.2 Defining Global Script Pre- and Post-Actions

Use this procedure to define an action that you want to execute before the first script panel appears (a pre-action) or after the last panel is executed in a script (a post-action).

Prerequisites

- To define a global script pre- or post-action for a wizard script, you must first graph the script.
- Create or open a graphical script. You must have a script open to designate global pre- and post-actions.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

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

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

   The Command window appears.

4. Define a command for the action. (See Defining Commands.)

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

See Also

- Designating Global Script Properties
- Defining the Script Information Area
- Defining Shortcut Buttons
- Programming the Script Disconnect Button

4.7.3 Defining the Script Information Area

The script information area can contain up to nine data elements (each either a string of text or a timer). Each data element may also contain an optional label.

The script information area displays in three rows, each able to display a data element and label at the left, middle and right position. The data elements can be any combination of alphanumeric elements or timers.

Sometimes referred to as the Static Information Panel, the data elements here actually may contain either static or dynamic information. This area is often used to identify static information such as the particular campaign name or business purpose of the script. It may also be used dynamically, displaying dynamic timers or customer profile information. This can be populated by executing a query (using a block) for customer information, and, using commands associated from the information area in Script Author, populating the data elements in the script area at runtime.

Using Scripting APIs, timers can be started, stopped, and resumed; and data in the script information area can be updated by explicit command.

Use this procedure to set up the information to appear in the script information area for the Scripting Engine agent interface.

Prerequisites

- To enable this feature for a wizard script, you must first graph the script.

- Create or open a graphical script. You must have a script open to define script information area data elements.

Steps

1. Access the graphical script's global properties using one of the following methods:
   - Select **File > Script Properties**, or
   - In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

   The Properties window appears.
2. In the Script tree, select **Static Panel**.
3. In the Static Panel pane, click an area in the representation of the script header where you would like this data element to appear.
4. If you want to insert text into the script information area, then click **Text**.
5. If you want to insert a timer into the script information area, then click **Timer**.
6. In the ID field, enter an identifying value.

   The Oracle Scripting API requires this ID.
7. Optionally, in the Label field, enter the label that you want to appear in the script information area at runtime, to label the designated data element.
8. In the Command field, click **Edit**.

   The Command window appears.
9. Define a command for the information panel text or timer.

   For general information regarding defining commands, see Defining Commands.
10. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

Guidelines

- If the data element you add to the script information area is a text field, define a constant command, with the value you want to appear at runtime designated as the return value of the command.

- If the data element you add to the script information area is a timer, define a Java method that invokes the startTimer Oracle Scripting API.
- For more information, see *Oracle Scripting Developer's Guide*.

See Also

- Designating Global Script Properties
- Defining Global Script Pre- and Post-Actions
- Defining Shortcut Buttons
- Programming the Script Disconnect Button

4.7.4 Defining Shortcut Buttons

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 as a global property of the script, the shortcut button area is empty at runtime. If shortcut buttons are defined, they appear in this area.

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.

The function, display content, and tool tips for shortcut buttons are global script properties. Button properties are configured using the Shortcut Panel in the Script Author script properties window.

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.

Use this procedure to define a shortcut button in the toolbar of the Scripting Engine agent interface.

Prerequisites

Insert a group.

Steps

1. Define the destination for the shortcut button by doing the following:
   a. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
   b. On the canvas, double-click a group.

      The Properties window appears.
   c. In the Group tree, select **Shortcut**.
   d. In the Shortcut pane, in the Shortcut field, type the name of the shortcut.
   e. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window for the group.
2. Define the shortcut button by doing the following:
   a. Access the graphical script's global properties using one of the following methods:
      * Select **File > Script Properties**, or
      * In Toggle select mode, right-click an empty portion of the canvas and select **Edit Blob Properties**.

      The Properties window appears.
   b. In the Script tree, select **Shortcut Panel**.

      The Shortcut Panel pane appears.
   c. In the Shortcut Panel pane, select **Add**.

      The Shortcut Info Entry Dialog window appears.
   d. In the ID field, type the shortcut name for the group.
   e. In the Label field, type the label that appears on the shortcut button at runtime.
   f. In the Tooltip field, type an on screen description of the shortcut button that appears when the user's pointer pauses over the button.
   g. Click **Edit** in the Command field.

      The Command window appears.

h. Define a command for the shortcut button. (See Defining Commands.)

i. If you want to make the shortcut enabled in the script interface for the compiled script, then select **Enabled**.

j. Click **OK** to exit the Shortcut Info Entry Dialog window.

k. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window for the script.

References

- Defining Commands
- Shortcuts
- For more information on using and defining shortcuts, see *Oracle Scripting Developer's Guide.*

See Also

- Designating Global Script Properties
- Defining Global Script Pre- and Post-Actions
- Defining the Script Information Area
- Programming the Script Disconnect Button

4.7.5 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.
- 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

See Also

- Designating Global Script Properties
- Defining Global Script Pre- and Post-Actions
- Defining the Script Information Area
- Defining Shortcut Buttons

4.8 Defining 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.

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 Corporation 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.

Panels may include any number of questions (there are no limits other than practicality).

You can perform the following tasks:

- Inserting a Panel
- Defining Panel Properties
- Defining Panel Question UI Controls
- Defining Panel Text and Layouts

See Also

- Getting Started with Script Author

- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.8.1 Inserting a Panel

Use this procedure to insert a panel onto the canvas. 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 more information on panels, see Defining Panels.

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Panel Insertion Mode** tool.

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

2. On the canvas, click where you want to insert the panel.

   If the Popup On Blob Creation option is selected, then the Properties window for the object appears.

> **Note:** 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).

References

- Defining Panel Question UI Controls
- Working with Questions
- Setting Properties Window to Pop Up at Object Creation
- Minimum Requirements for Any Graph
- Inserting a Block
- Inserting a Group
- Inserting a Termination Node

See Also

- Defining Panel Properties
- Defining Panel Question UI Controls
- Defining Panel Text and Layouts

4.8.2 Defining Panel Properties

Use this procedure to define the properties of a panel. Panel properties affect the way in which a panel is stored in the database and displayed in runtime.

You can perform the following tasks:

- Defining the Panel Name
- Defining Panel Comments
- Defining the Panel Label
- Opening the Panel Layout Editor

- Defining Panel Text and Layouts
- Substituting a Java Bean for a Panel

See Also

- Inserting a Panel
- Defining Panel Question UI Controls
- Defining Panel Text and Layouts

4.8.2.1 Defining the Panel Name

The panel name is used to identify each individual panel in Script Author and in the database. Script Author uses the panel name value to track panels visited in a script session for footprinting purposes.

Note that the panel name is distinct from the panel label, which is displayed in the Progress Area of the Scripting Engine agent interface at runtime.

Use this procedure to define the name of a panel.

Prerequisites

Insert a panel onto the canvas.

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, in the Name field, type the name of the panel.

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

See Also

- Defining Panel Comments
- Defining the Panel Label
- Opening the Panel Layout Editor
- Defining Panel Text and Layouts
- Substituting a Java Bean for a Panel

4.8.2.2 Defining Panel Comments

Panel comments are optional. They provide a convenience for script developers, and may be left blank. You may choose to include information in the panel comments field that is helpful to other script developers. There is no maximum character limit to a comment and no character restrictions.

Use this procedure to define a comment for a panel.

Prerequisites

Insert a panel onto the canvas.

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, in the Comments field, type the desired comment.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining the Panel Name
- Defining the Panel Label
- Opening the Panel Layout Editor
- Defining Panel Text and Layouts

- Substituting a Java Bean for a Panel

4.8.2.3 Defining the Panel Label

The panel label is the value used to identify specific panels in the Progress Area of the Scripting Engine agent interface at runtime. Only panels that have been visited by the end user in the flow of that script session will be listed (by panel label) in the Progress Area.

Use this procedure to define the label for a panel.

Prerequisites

Insert a panel onto the canvas.

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 Label field, type the label of the panel.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining the Panel Name
- Defining Panel Comments
- Opening the Panel Layout Editor
- Defining Panel Text and Layouts
- Substituting a Java Bean for a Panel

4.8.2.4 Opening 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.

Use this procedure to open the panel layout editor.

Prerequisites

Insert a panel onto the canvas.

Steps

1. If you want to open the panel layout editor using the toolbar, perform the following steps:
   a. In the Object and Branch toolbar, click the Toggle Select Mode tool.
   b. On the canvas, click the target panel.
   c. From the toolbar, click **Panel Layout**.

      The Panel Layout Editor window appears.
2. If you want to open the panel layout editor from the panel properties window, perform the following steps:
   a. In the Object and Branch toolbar, click the **Toggle Select Mode** tool.
   b. On the canvas, double-click a panel.

      The Properties window appears.
   c. In the Panel tree, select **Panel Layout**.

      The Panel Layout pane appears.
   d. In the Panel Layout pane, click **Panel Layout Editor**.

      The Panel Layout Editor window appears.

See Also

- Defining the Panel Name
- Defining Panel Comments
- Defining the Panel Label
- Defining Panel Text and Layouts
- Substituting a Java Bean for a Panel

4.8.2.5 Defining Panel Text and Layouts

You can define panel text and layouts using the panel layout editor.

References

- Inserting a Panel
- Defining Panel Properties
- Defining Panel Question UI Controls

See Also

- Defining the Panel Name
- Defining Panel Comments
- Defining the Panel Label
- Opening the Panel Layout Editor
- Substituting a Java Bean for a Panel

4.8.2.6 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.
- 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

See Also

- Defining the Panel Name
- Defining Panel Comments
- Defining the Panel Label
- Opening the Panel Layout Editor
- Defining Panel Text and Layouts

4.9 Defining Groups

You can perform the following tasks:

- Inserting a Group
- Importing a Saved Script as a Group
- Defining or Changing the Group Name
- Defining Shortcuts

References

- Defining Group Pre- and Post-Actions

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.9.1 Inserting a Group

Use this procedure to insert a group onto the canvas.

Note: Inserting a group creates a subgraph that must adhere to the minimum requirements for any graph. Ensure you define a termination node in the group subgraph and include branching from the start node to the termination node, including branching to any other objects as appropriate.

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Group Insertion Mode** tool.

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

2. On the canvas, click where you want to insert the group.

   The **Go down into child graph** tool on the Navigation and Alignment toolbar enables, indicating the presence of the subgraph created by the group.

   If the Popup On Blob Creation option is selected, then the Properties window for the object appears.

3. Define properties for the inserted group, as appropriate.

   - To define a group name or comments for the group, see Defining or Changing the Group Name.
   - To define a shortcut, see Defining Shortcuts.

References

- Setting Properties Window to Pop Up at Object Creation
- Inserting a Panel
- Inserting a Block
- Inserting a Group
- Inserting a Termination Node
- Drilling Down Into or Up From a Group or Block
- Defining Global Script Pre- and Post-Actions

See Also

- Importing a Saved Script as a Group
- Defining or Changing the Group Name
- Defining Shortcuts

4.9.2 Importing a Saved Script as a Group

When you import a script, the entire script (the root graph, and all sub-graphs) represented by that script is placed into your current script, within a group. By default, the imported group is named using the global properties of the imported script.

This procedure is described in the task called Importing a Script.

References

- Importing a Script

See Also

- Inserting a Group
- Defining or Changing the Group Name
- Defining Shortcuts

4.9.3 Defining or Changing the Group Name

The group name is the property used to reference a group visually on the canvas. It is defined in the properties window for the group.

Use this procedure to define or change the name of a group.

Prerequisites

Insert a group onto the canvas.

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 **Properties**.

   The Properties pane appears.
4. In the Properties pane, in the Name field, type the name of the group.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Inserting a Group
- Importing a Saved Script as a Group
- Defining Shortcuts

4.9.4 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.

   The Properties window appears.
3. In the Group tree, select **Shortcut**.

   The Shortcut pane appears.
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 (at minimum) a default branch from the start node to a termination node.

References

- Defining Shortcut Buttons
- Defining Commands
- Inserting a Termination Node

See Also

- Inserting a Group
- Importing a Saved Script as a Group
- Defining or Changing the Group Name

4.10 Defining 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. The object dictionary of the block allows you to specify your database function in a simple tab-based user interface.

Using an API block, you can add any Script Author command (Java, PL/SQL, blackboard, forms, constant, or a delete action) to your script. The block gives a clear visual indicator to the script developer that data source integration occurs at that point in the script.

Context Sensitive Object Dictionary

By defining a block type and completing information in the object dictionary of the block, you identify parameters for your SQL action or command.

The object dictionary includes three tabs, for defining:

- Which tables or views are accessed by the block.
- If accessing multiple tables, the join conditions for the SQL action.
- Constraints to the SQL action.

The object dictionary is context sensitive to the block object type.

- When you define a block type as an API block, the object dictionary lists API actions (hooks for a Script Author command). From here you can access the command library (to insert a previously defined command) or define a custom command (including referencing best practice Java commands).
- When you define a block type as either an insert block or an update block, the third tab in the object dictionary is labeled Data Constraints. You can define a command to specify the constraint for the SQL action. This command includes a data dictionary for the constraint.

- When you define a block type as either a query block, the third tab in the object dictionary is labeled Query Constraints. This includes a tab to specify the columns queried, and a tab to define the "WHERE" clause of your select query. If your query requires multiple conditions, you can define each constraint as a separate constraint, or you can define one constraint, including AND or OR statements as applicable.
- If performing a query as a prerequisite to a subsequent update, this option is accessible from the Query Data tab of a query block.

For information on using these features, see the respective task in this section.

Database Connection Information

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).

For any block from which you are performing a SQL action, you can then select the appropriate connection from the Connection/Tables list. Connection options in that window include:

- **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.

For more information on database connections specific to blocks, see Defining Database Connections for a Block. For information specific to database connections in general, including how to test database connections, see Defining Database Connections.

You can perform the following tasks:

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block

- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

References

- Defining Block Pre- and Post- Actions
- Defining Actions
- Defining Commands
- Defining Block API Actions

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls

- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.10.1 Inserting a Block

Use this procedure to insert a block onto the canvas.

You can then define the function of the block by selecting the block type and specifying details in the object dictionary.

> **Note:** Inserting a block creates a subgraph that must adhere to the minimum requirements for any graph. Ensure you define a termination node in the block subgraph and include branching from the start node to the termination node, including branching to any other panels contained in the subgraph, as appropriate.

Prerequisites

None

Steps

1. In the Object and Branch toolbar, click the **Block Insertion Mode** tool.

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

2. On the canvas, click where you want to insert the block.

   A block is placed on the canvas. A generic name such as Block4 is automatically provided to the block.

   The **Go down into child graph** tool on the Navigation and Alignment toolbar is made active.

   If the Popup On Blob Creation option is selected, then the Properties window for the object appears.

References

- Setting Properties Window to Pop Up at Object Creation

- Inserting a Panel
- Inserting a Group
- Inserting a Termination Node

See Also

- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.2 Defining the Block Name

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

Prerequisites

Insert a block onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a block.

   The Properties window appears.
3. In the Block tree, select **Properties**.

   The Properties pane appears.

4. In the Properties pane, in the Name field, type the name of the block.
5. Optionally (for example, if you want to provide a note regarding the function of the block or details about its creation), in the Comments field, type a comment.
6. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.3 Defining a Database Query Block

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 a database table update using an update block.

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 first record in the cursor can also be displayed in panel text (as an embedded value containing a blackboard command). 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 block is only stored in the cursor until a subsequent query is executed, or until the script session is completed. 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.
- 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.

Information returned by a query is stored and accessed by its table and column name, in format TABLE.COLUMN.

Use this procedure to define a block as a query block.

Prerequisites

- If you want to query a table or view in a database instance that is different than the instance you use for Script Author, you must first define the new database connection.
- Insert a block onto the canvas.
- In addition, you need the following information:
  - The names of the database tables or views that contain the information that you want to search or retrieve.
  - The names of the table columns that contain the search criteria and that contain the information that you want to retrieve.
  - The data types of the appropriate table columns.
  - If joining two or more tables or views, the names of the primary keys and, if any, the foreign keys of the tables.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.

2. On the canvas, double-click a block.

   The Properties window appears.

3. In the Block tree, select **Types.**

   The Types pane appears.

4. In the Types pane, from the Block Types list, select **Query Block.**

5. In the Block tree, select **Object Dictionary.**

   The Object Dictionary pane appears.

6. In the Connection/Tables tab, do the following:

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

   b. List the database tables that contain the information that you want to search or retrieve. (See Defining Database Tables for a Block.)

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

8. In the Query Data tab, do the following as appropriate:

   a. Optionally, in the Query Data area, in the Prefetch Value field, type 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.

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

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

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

   d. Using the Query Constraints tab, define your search criteria. (See Defining Query Constraints for a Query Block.)

   e. Using the Query Columns tab, list the information that you want to retrieve. (See Defining Query Columns for a Query Block.)

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

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.4 Defining a Database Insert Block

Using an insert block, you can gather information and write records to one or more database tables specified within the data dictionary of the block.

First, insert a block into the script, and then define the block type and data dictionary properties to perform the insert action.

Inserting a block into a script creates a subgraph. If desired, you can add Script Author objects into the subgraph which perform the gathering of the information required for the insert action. For example, if creating a customer record, you can add panels with questions that collect the customer name, title of address, and so on.

Each instance of executing the insert block at runtime results in the insertion of a single record in the specified database tables.

Any objects contained within the block are processed before the block's database function (in this case, the insertion of records into the specified database tables) executes at script runtime.

You must ensure that all data required to be inserted into the database is obtained prior to executing the block. The database function of the block occurs when the termination node within the block is reached, followed by any post-actions defined for the block.

One common method to ensure all data is available is to define validation 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.

If all the information required for the insert action in a script is already known, you may have no requirements for Script Author objects within the block subgraph. In this case, you must still adhere to the minimum requirements for any graph; thus, define a termination node in the block subgraph and use a default branch from the start node to the termination node.

Use this procedure to define a block as an insert block to insert a record into a database.

Prerequisites

- If you want to insert a record in a table or view in a database instance that is different than the instance you use for Script Author, you must first define the new database connection.
- Insert a block onto the canvas.
- In addition, you need the following information:
  - The names of the database tables or views into which you want to insert a record.
  - The names of the table columns into which you want to insert a record.
  - The data types of the appropriate table columns.
  - If joining two or more tables or views, the names of the primary keys and, if any, the foreign keys of the tables.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.

2. On the canvas, double-click a block.

   The Properties window appears.

3. In the Block tree, select **Types.**

   The Types pane appears.

4. In the Types pane, from the Block Types list, select **Insert Block**.

5. In the Block tree, select **Object Dictionary.**

6. In the Connection/Tables tab, do the following:

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

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

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

   Note: If you are joining multiple tables, then you must indicate the primary keys for each table.

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

   a. Drill down into the block

   b. Insert a panel in the block.

   c. In the Properties pane of the designated panel, define a panel name.

   d. In the Answers pane of the designated panel, define a a question. (See Defining Panel Question UI Controls.)

   e. From the Answer Entry Dialog window, click **Edit Data Dictionary**.

      The Edit Data Dictionary dialog appears. The default data dictionary name is untitledDataDictionary[number].

   f. Click the **Table Info** tab.

   g. In the Table field, type the name of the table into which you want the answer inserted.

h. In the Column field, type the name of the column into which you want the answer inserted.

i. Click **OK** to exit the Data Dictionary window.

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

---

j. Click **OK** to exit the Answer Entry Dialog window.

k. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

l. Insert a termination node onto the canvas.

m. Draw branches between the objects in the block subgraph to indicate the flow of the script.

n. Save your work in progress by selecting **File > Save**.

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

a. In the Block tree of the insert block, **Object Dictionary.**

The Object Dictionary pane appears.

b. In the Object Dictionary pane, click the Data Constraints tab.

The Data Constraints area appears.

c. In the Data Constraints area, click **Add**.

The Data Constraint window appears.

d. In the Command area, click Edit.

The Command window appears.

e. Define a command that obtains or derives the value to be inserted. (See Defining Commands.) Save the command and close this window.

f. In the Data Constraint window, in the General area, in the Name field, type the name of the data constraint.

Note: After the command is executed, the value is stored on the Oracle Scripting blackboard under the name of the data constraint. This value, therefore, is also referred to as a blackboard key.

g. In the Data Constraint window, click the Table info tab.

h. In the Table data area, in the Table field, type the name of the table into which the value is inserted.

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

j. Click **OK** to exit the Data Constraint window.

k. If you want to add another data constraint, then repeat steps a through j.

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

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.5 Defining a Database Update Block

You can update existing records in a database table or view using the Update block. Like the other database functions, include the database details (tables, columns, constraints and join information, if relevant) in the block data dictionary.

Each update block in a script must be precluded by a query block which retrieves at least one valid row in the table in which you subsequently perform the update.

Use this procedure to define a block in a script that updates a record in a database.

Prerequisites

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

  Note: Oracle Corporation recommends including code in your script prior to an update block to validate that the prerequisite query returns at least one valid row.

- If you want to update a table or view in a database instance that is different than the instance you use for Script Author, you must first define the new database connection.
- Insert a block onto the canvas.
- In addition, you need the following information:
  - The names of the database tables or views into which you want to update a record.
  - The names of the table columns into which you want to update a record.
  - The data types of the appropriate table columns.
  - If joining two or more tables or views, the names of the primary keys and, if any, the foreign keys of the tables.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a block.

   The Properties window appears.
3. In the Block tree, select **Types.**

   The Types pane appears.
4. In the Types pane, from the Block Types list, select **Update Block.**
5. In the Block tree, select **Object Dictionary.**

6. In the Connection/Tables tab, do the following:
   a. Define how Oracle Scripting connects to the database at runtime. (See Defining Database Connections for a Block.)
   b. List the database tables that contain the information that you want to update. (See Defining Database Tables for a Block.)
7. If you want to update data in more than one database table, then, in the Join Condition tab, indicate the relationship between related tables. (See Defining Join Conditions for a Block.)

   Note: If you are joining multiple tables, then you must indicate the primary keys for each table.

8. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.
9. If you want to use the script end user's response to a panel question as the value with which to update a database table, then do the following:
   a. Drill down into the block
   b. Insert a panel in the block.
   c. In the Properties pane of the designated panel, define a panel name.
   d. In the Answers pane of the designated panel, define a a question. (See Defining Panel Question UI Controls.)
   e. From the Answer Entry Dialog window, click **Edit Data Dictionary**.

      The Edit Data Dictionary dialog appears. The default data dictionary name is untitledDataDictionary[number].
   f. Click the **Table Info** tab.
   g. In the Table field, type the name of the table to be updated.
   h. In the Column field, type the name of the column to be updated.
   i. Click **OK** to exit the Data Dictionary window.

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

j. Click **OK** to exit the Answer Entry Dialog window.

k. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

l. Insert a termination node onto the canvas.

m. Draw branches between the objects in the block subgraph to indicate the flow of the script.

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

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

The Data Constraint window appears.

b. In the Command area, click Edit.

The Command window appears.

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

d. In the Data Constraint window, in the General area, in the Name field, type the name of the data constraint.

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

---

e. In the Data Constraint window, click the Table info tab.

f. In the Table data area, in the Table field, type the name of the table to be updated.

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

h. Click **OK** to exit the Data Constraint window.

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

11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Inserting a Block

- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.6 Defining Database Connections for a Block

Oracle Scripting can easily communicate with database tables or views of your choosing. When performing database operations using a block, you must specify the connection intended for use during script execution.

To connect to a database table within the same database instance that you log into when starting the Oracle Applications session to use the Script Author applet, you must specify that the application use the login connection. The appropriate database is then referenced at runtime.

If connecting to a database table or view associated with a database instance that is different than the one used when you log into Oracle Applications for your current Script Author session, you must define new connection parameters using the Connection List.

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

Prerequisites

Insert a block onto the canvas.

Steps

1. If you want to connect to a database table or view using the same database instance as the Script Author applet, then do the following:
   a. In the Object and Branch toolbar, click the Toggle Select Mode tool.

b. On the canvas, double-click a block.

The Properties window appears.

c. In the Block tree, select **Object Dictionary.**

d. In the Connection/Tables tab, select **Use Login Connection.**

e. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

2. If you want to connect to a database table or view in a database instance that is different than the one used when you log into Oracle Applications for your current Script Author session, then from the Script menu, select **Connection List**.

The **Connection List** window appears.

a. If you want to edit an existing connection, in the Connection List window, in the Connection List area, click **Edit**.

A warning appears, indicating that any objects using this connection will be affected by the change, and asking if you want to edit the connection.

* To cancel this request and return to the Connection List window, click **No**.
* To proceed, click **Yes**.

The Connection window appears.

b. If you want to add a new connection, click **Add.**

The Connection window appears.

c. In the Connection window, select the **Connection** tab.

d. In the Connection tab, in the Connection Name field, type a meaningful name.

This name identifies this connection and will subsequently appear in the Connection list.

e. In the Host Name field, type the appropriate database host, including domain if appropriate.

f. In the Port Number field, type the port number of the database instance to which you want to connect.

g. In the SID field, type the SID of the database instance to which you want to connect.

h. In the User Name field, type the username of an Oracle Applications user with apps level access to the appropriate database instance.

   For example, type apps.

i. In the Password field, type the appropriate password for this user.

j. To save your connection information, click **OK**.

   The Connection window closes. The Connection List window appears.

3. If you want to test a defined database connection, then do the following:

   a. From the Script menu, select **Connection List**.

      The Connection List window appears. Each configured connection appears in the list.

   b. Select the connection you want to test and click **Edit**.

      A warning appears, indicating that any objects using this connection will be affected by the change, and asking if you want to edit the connection.

   c. In the warning window, click **Yes**.

      The Connection window appears.

   d. Select the Test tab.

   e. Click the test button.

      View the status of the connection in the Status area.

   f. To exit, click **OK** or **Cancel**.

      The Connection window closes. The Connection List window appears.

   g. To exit the connection list, click **Close.**

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Tables for a Block

- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.7 Defining Database Tables for a Block

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

Prerequisites

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

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a block.

   The Properties window appears.
3. In the Block tree, select **Object Dictionary.**
4. In the Connection/Tables tab, in the Tables area, click **Add table**.

   The Table window appears.

   Note: The Table window can be resized to extend the viewable area. To do so, place your cursor on either side of the Table window until it changes to a bi-directional arrow. Then, click the mouse button, and drag to stretch the window to the appropriate size. The Table window will appear at its default size each time it is opened.
5. Type the name of the table and then click **OK** to exit the Table window.
6. If you want to add another table, then repeat steps 4 through 5.

Note: List master tables first and then tables related by foreign key.

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

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.8 Defining Join Conditions for a Block

When you need to access more than one database table or view to perform a SQL action in a script, use the Join Condition tab in the block object dictionary to indicate the links between related tables.

Tables are joined according to common values existing in corresponding columns.

Note: If you are joining multiple tables, then you must indicate the primary keys for each table.

You can also provide detail information for any primary key and foreign key tables.

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

Use this procedure to indicate the links between related tables when defining a SQL function in a block.

Prerequisites

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

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a block.

   The Properties window appears.
3. In the Block tree, select **Object Dictionary.**

   The Object Dictionary pane appears.
4. In the Object Dictionary pane, click the **Join Condition** tab.

   The Join Condition area appears.
5. In the Join Condition tab, click **Add**.

   The Join Condition Dialog window appears.
6. In the Master PK Table tab, do the following:
   a. In the Master Table field, type the name of the table containing the relevant primary key.
   b. Click **Add Master PK**.

      The Column Entry window appears.
   c. Type the name of the column that is the primary key of the master table, and then click **OK** to exit the Column Entry window.

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

The Column Entry window closes.

The primary key you entered appears in the Master PK Table list.

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

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

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

b. Click **Add Detail PK**.

The Column Entry window appears.

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

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

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

a. Click **Add Detail FK**.

The Column Entry window appears.

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

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

---

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

9. Click **OK** to exit the Join Condition Dialog window.

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

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block

- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.9 Defining Data Constraints for an Insert or Update Block

Blocks are the visual method Oracle Scripting employs to allow you to perform a SQL action or include an API in your script. When viewing the script layout on the canvas, the block serves as a clear visual indicator that one of these functions is included in the script.

By defining a block type and completing information in the object dictionary of the block, you identify parameters for your SQL action or command.

A block also serves as a container, creating a subgraph that may contain one or more panels. Panels within a block allow the script to collect answers used as parameters for the block action. The answers to these panels at runtime can serve as constraints to the SQL function. As soon as the termination node within a block is reached at runtime, the function of the block is executed. In this way, if panels within the block collect information, the responses provided to those panel questions at runtime are available and used to provide constraints for the SQL or API function. For information on defining data constraints using panels, see Defining Panels within a Block.

When you define a block type as either an insert block or an update block, the object dictionary includes a Connection/Tables tab, a Join Condition tab, and a Data Constraints tab.

For insert and update SQL actions, use the Data Constraints tab to define a command to provide data constraints for the SQL action.

- You can define a Script Author PL/SQL command to obtain information from a database table or view to use as a parameter for the insert or update action.
- You can define a blackboard command to obtain information from the blackboard to use in the insert or update action.
- You can define a Java command to obtain the required data to use in the insert or update action.

- You can insert a forms command to obtain the required data to use in the insert or update action from an open form. Not all forms-based applications allow Oracle Scripting to obtain information from the form.

Use this procedure to define data constraints for rows to be inserted or updated.

Prerequisites

- Insert a block onto the canvas.
- In addition, you need the names of the database tables and columns that contain the information that you want to update or insert.
- In addition, you need to know the full construction of the WHERE clause and whether the desired data constraint values derive from a command or the cursor.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click an insert or update block.

   The Properties window appears.
3. In the Block tree, select **Object Dictionary.**
4. Select the **Data Constraints** tab.
5. In the Data Constraints area, click **Add**.

   The **Data Constraint** window appears.
6. Define the data constraint.

   Note: Conditions based on user input parameters require a panel in the block to gather the input. The *value* is a question name in the panel. See Defining Panels within a Block.

   a. To define a command as a data restraint, click **Edit** and define the command. For more information, see Defining Commands.

   b. To define a database table (required when using a table lookup as a data restraint), in the Data Constraint window, click the **Table Info** tab and in the Table Data area, specify the database table and column name.

Optionally, specify the format mask. To improve performance, specify the data type.

c. To define a table, cursor, or command lookup as a data restraint, in the Data Constraint window, click the **Lookups** tab and in the Lookups area, specify the lookup table name, cursor parameters, or command, as appropriate.

Click **OK** to save your work and exit the Data Constraints window.

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

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.10 Defining Query Constraints for a Query Block

By defining a block type and completing information in the object dictionary of the block, you identify parameters for your SQL action or command. For query blocks, the object dictionary includes a Connection/Tables tab, a Join Condition tab, and a Query Constraints tab. Use the Query Constraints tab to define constraints for your query.

A block also serves as a container, creating a subgraph that may contain one or more panels. Panels within a block allow the script to collect answers used as parameters for the block action. The answers to these panels at runtime can serve as constraints to the SQL function. As soon as the termination node within a block is reached at runtime, the function of the block is executed. In this way, if panels

within the block collect information, the responses provided to those panel questions at runtime are available and used to provide constraints for the SQL or API function. For information on defining data constraints using panels, see Defining Panels within a Block.

Use this procedure to refine your search criteria by defining query constraints for rows to be selected from the database.

For additional guidelines for defining query constraints, with specific examples, see the query constraint examples section.

Query Constraints Examples

EXAMPLE 1:

Assume you have obtained a customer number (12345) using a query or PL/SQL action earlier in the script, and this number is available to the Scripting Engine. You can then define a data constraint in a block equivalent to the SQL clause WHERE CUSTOMER_NUMBER = 12345, using a single constraint entry.

EXAMPLE 2:

Assume you have obtained a customer's first and last name (Nelson Mandela) using a query or PL/SQL action earlier in the script, and this data is available to the Scripting Engine. You can then define constraints in a block equivalent to the SQL clause WHERE FIRST_NAME = 'NELSON' AND LAST_NAME = 'MANDELA', using two separate constraint entries. Alternatively, you can include the AND statement in your block syntax and identify both constraints in the same entry.

Data Volatility

Your approach to accomplish either of the prior examples differs based on from where you obtain the data you use for the constraint.

If you perform a query, this information is available in the scripting cursor (and by extension, to the scripting blackboard) only until the next query is executed. Any subsequent query will replace existing values in the cursor.

> **Note:** For most effective queries, when referencing the column name, prefix the table name (for example, *table1.column1*).

If, however, your script includes a command following the query to write the information to the blackboard, then the queried information is available from that point forward in the current script transaction. If, in your command, you choose to

rename the blackboard key, you must reference the blackboard key name in your constraint. In this case, you need not specify the table name; simply use the blackboard key you defined. These values are case-sensitive.

Assume for these examples that you have a custom table called CUSTOMER_INFO, and in this table is a column called PhoneNbr.

EXAMPLE 3

If you have the phone number for customer 12345 from a query, and wish to query another database table, using as a constraint the condition in which the phone number in the database matches the one in the cursor, then use as the string within the Query Constraint Input window the following:

CUSTOMER_INFO.PhoneNbr =: PhoneNbr

Include a space both after the column identifier (CUSTOMER_INFO.PhoneNbr) and before the cursor and blackboard value (PhoneNbr).

EXAMPLE 4

Assume that, after the query in which you obtained the phone number for customer 12345, you referenced in a Java command (as a post-action to the block) the following Java method, using the setBlackBoardAnswer API, to set returned values to the blackboard:

```
public void setBlackboard(ServerProxyBean pb, String key, String value)
    {
        try
        {
            pb.setBlackBoardAnswer(key, value);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
```

Further, assume that you defined a blackboard key (PHONE_NUMBER) for the table value CUSTOMER_INFO.PhoneNbr.

Thereafter, in that session of the script, you can obtain that phone number using the blackboard key PHONE_NUMBER.

Also, you can now use this in your constraint. Using the same case as in example 3, if you then wish to query another database table, using as a constraint the condition

in which the phone number in the database matches the one in the blackboard, then use as the string within the Query Constraint Input window the following:

CUSTOMER_INFO.PhoneNbr =: PHONE_NUMBER

For example, if the phone number in the blackboard is 650-555-7777, then this constraint results in a WHERE clause of WHERE CUSTOMER_INFO.PhoneNbr =: 650-555-7777.

Prerequisites

- Insert a block onto the canvas.
- In addition, you need the names of the database tables and columns that contain the information that you want to query.
- In addition, you need to know the full construction of the WHERE clause and whether the desired query constraint values derive from a command or the cursor.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click an insert or update block.

   The Properties window appears.
3. In the Block tree, select **Object Dictionary.**
4. Select the **Query Data** tab.
5. In the Query Data area, in the Query Constraints tab, click **Add Constraint**.

   The **Query Constraint Input** window appears.
6. Define the query constraint.

   Note: Conditions based on user input parameters require a panel in the block to gather the input. The *value* is a question name in the panel. See Defining Panels within a Block.

   The following examples correspond the query constraint examples used earlier.

   a. To set a constraint in which the customer number column in the customers table is equal to 12345, in the Input a string field, type CUSTOMERS.CUSTOMER_NUMBER =: 12345 and click **OK**.

b. To set a constraint in which the customer last name column in the customers table is equal to Nelson, in the Input a string field, type CUSTOMERS.LAST_NAME =: 'Nelson' and click **OK**. Create a second constraint of where the customer last name column in the customers table is equal to Mandela, type CUSTOMERS.LAST_NAME =: 'Mandela' and click **OK**.

c. This example presupposes that you have a customer phone number in the scripting cursor from a previous query. To set a constraint in which you are obtaining the customer number, first and last names from the CUSTOMERS table where the customer phone number in the CUSTOMER_INFO table is equal to the value in the scripting cursor, in the Input a string field, type CUSTOMER_INFO.PhoneNbr =: PhoneNbr and click **OK**.

d. This example presupposes that you have obtained a customer phone number from the PhoneNbr column from the CUSTOMER_INFO table, and have executed a command to set this value to the blackboard using the blackboard key PHONE_NUMBER. To set a constraint in which you are obtaining the customer number, first and last names from the CUSTOMERS table where the customer phone number in the CUSTOMER_INFO table is equal to the PHONE_NUMBER value in the scripting blackboard, in the Input a string field, type CUSTOMER_INFO.PhoneNbr =: PHONE_NUMBER and click **OK**.

When you click **OK**, the Query Constraint Input window closes, and any constraint you added appears in the Query Constraint list in the Query Constraints tab.

7. If you want to add another query constraint, then repeat steps 5 through 6.

Note: Insert logical operators (such as, NOT, AND, and OR) at the end of the query constraint. If no logical operator is used, then AND is assumed. Use parentheses at the beginning or end of a query constraint to override the rules of precedence.

---

8. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Inserting a Block
- Defining the Block Name

- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Columns for a Query Block
- Defining Panels within a Block

4.10.11 Defining Query Columns for a Query Block

Use this procedure to specify the columns from which you want to retrieve data from the database.

Prerequisites

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

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a query block.

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

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

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

6. If you want to improve network performance during the query, then select the type of data in the column from the Data Type list.
7. Click **OK** to exit the Enter a Key/Value window.
8. If you want to add another column, then repeat steps 4 through 7.
9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Panels within a Block

4.10.12 Defining Panels within a Block

A block provides a script with the mechanism to perform a database function (insert, select, or update), or to call a Script Author command as an API action. A block also serves as a container, creating a subgraph that may contain one or more panels. Panels within a block allow the script to collect answers used as parameters for the block action. Thus, for queries (select statements), inserts, or updates, the answers collected at runtime can serve as constraints to the SQL function.

Like any graph in a script, the subgraph within each block automatically contains a start node, must be connected from that start node with a default branch to the next object in a script, and must contain a termination node on the graph.

Blocks also allow one or more panels to be included in the subgraph. All objects require appropriate branching. If no panels are required, a syntactically correct

block subgraph is comprised of the start node, connected with a default branch to a termination node.

As soon as the termination node within a block is reached at runtime, the function of the block is executed. In this way, if panels within the block collect information, the responses provided to those panel questions at runtime are available and used to provide constraints for the SQL or API function.

Blocks used for SQL functions are also capable of defining constraints outside of the context of the panel. For information known prior to execution of the script, this information can be included in the block's object dictionary. The third tab in the object dictionary for a SQL function (insert, query, or update) block can be used to identify constraints to the SQL to be executed.

Thus, constraints can be added to a SQL function using panels, the object dictionary, or both.

For example, you can include a panel in a query block that requests an account number. The account number is passed to the query, which performs a select statement to obtain transaction information.

Alternatively, if the account number is known, you can include it as a query constraint in the object dictionary of the query block, eliminating the panel.

Use this procedure to add a panel to a block.

> **Note:** A block creates a subgraph on the canvas. This subgraph must follow all the syntactical rules of any graph in a script. At minimum, it must contain a termination node and default branching between the start node and the termination node. Blocks may also contain panels. They may not contain groups or other blocks.

Prerequisites

Insert a block onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, click a block and then click the **Go down into child graph** tool on the Navigation and Alignment toolbar.

   The graph for the block appears.

3. In the Object and Branch toolbar, click the **Panel Insertion Mode** tool.

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

4. On the canvas, click where you want to insert the panel.

   If the Popup On Blob Creation option is selected, then the Properties window for the object appears.

5. Access the panel properties, and add (at minimum) a unique name for the panel, and at least one question.

6. Save your work.

7. Ensure appropriate branching from the panel to any other objects you add to the subgraph, including appropriate termination.

See Also

- Inserting a Block
- Defining the Block Name
- Defining a Database Query Block
- Defining a Database Insert Block
- Defining a Database Update Block
- Defining Database Connections for a Block
- Defining Database Tables for a Block
- Defining Join Conditions for a Block
- Defining Data Constraints for an Insert or Update Block
- Defining Query Constraints for a Query Block
- Defining Query Columns for a Query Block

4.11 Defining Branches

Branches directly control the flow of a script in runtime.

Each object in any graphical script that is to be processed at runtime is connected with one or more branches, starting from the automatically generated start node, and ending with the termination node (required on each graph to signal completion of processing for that graph). The only exception to this rule includes the use of indeterminate branches.

When a script is executed at runtime, the Scripting Engine directs its processing from object to object, primarily based on the business rules for the branch type or types associated with that object.

Script Author includes four branch types in a graphical script: default, distinct, conditional, and indeterminate branches. Each branch type enforces unique syntactical rules.

Each Script Author object to be processed includes one or more outgoing branches (branches originating from that object and leading to another on the canvas). Branches are evaluated in the order in which they are created. This default branch order can also be changed (using the Branches pane of the properties window for the relevant configurable object), resulting in a change in the order in which branches are evaluated at runtime.

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.

All branches except the Indeterminate branch are drawn from one object on the canvas to another. Indeterminate branches are drawn from one object on the canvas to an open point, since its destination is not known until its expression is evaluated and an object name or shortcut name is returned.

You can perform the following tasks:

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Default Branch
- Defining a Distinct Branch
- Defining a Conditional Branch
- Defining an Indeterminate Branch
- Defining the Branch Name
- Reordering Branches

References

- Inserting a Branch
- Defining Branches

- Working with Branches on the Canvas
- Controlling Script Flow
- Defining Actions
- Defining Commands

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.11.1 Setting Properties Window to Pop Up at Branch Creation

Use this procedure to set up Script Author to automatically display the Properties window when you insert a branch onto the canvas.

This display property, once set, is retained for the duration of the Script Author session.

Prerequisites

None

Steps

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

See Also

- Defining a Default Branch
- Defining a Distinct Branch
- Defining a Conditional Branch
- Defining an Indeterminate Branch
- Defining the Branch Name
- Reordering Branches

References

- Working with Branches on the Canvas

4.11.2 Defining a Default Branch

Use this procedure to insert a default branch from the source object to the destination object, which navigates the script at runtime to a default destination. A default destination is the destination for the flow of the script if no other branch types or conditions are used, or if any other conditions (as represented by other branch types) are not met at runtime.

Special Default Branch Rules

- The default branch can be used in combination with any or all other branch types from the same source object. Since branches are evaluated in order of branch creation, the default branch should be created as the last branch type created for a source object (or the branches for the given object should be reordered to designate the default branch as last). Any branch in a lower branch order than the default branch will not be evaluated at runtime.

- The default branch is used when there is only one destination, and no evaluation of logic is required.
- The default branch must be used from the start node (on the root graph and on any other graph in a script) to the first destination object.
- The default branch should be used to provide a default path any time the conditional branch is used, to ensure that the script can continue if the condition tested at runtime returns the boolean value "false."
- When using distinct branching from a panel, the default branch can be used to provide a default path for any answer choices not otherwise assigned a distinct destination. For example, if a question (Panel text "Pick a number from 1 to 4") has ten answer choices defined (the digits 1 through 10), and distinct branching is created to evaluate answers 1 and 2, then using a default branch to a third destination would result in the flow of the script if any number (from 3 through 10) is chosen at runtime.

Prerequisites

Two objects must exist on the canvas in order to connect them using the default branch.

Steps

1. In the Object and Branch toolbar, click the **Default Branch Mode** tool.

   When the pointer is over the canvas, the pointer is a crosshair (+).

2. On the canvas, drag the pointer from the source object to the destination object.

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

See Also

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Distinct Branch
- Defining a Conditional Branch
- Defining an Indeterminate Branch
- Defining the Branch Name
- Reordering Branches

References

- Working with Branches on the Canvas

4.11.3 Defining a Distinct Branch

A distinct branch determines the distinct path the Scripting Engine should take in a script at runtime when the script end user selects one distinct, predefined answer choice. This branch type requires answer choices to be defined in the data dictionary for a question.

Each distinct branch accounts for only one answer choice. You can define one distinct branch for each answer choice defined in the data dictionary for that question. Alternatively, if applicable, you can use more than one branch type from a source panel. Ensure the branch order is appropriately defined so that each condition that you want to include is evaluated.

Once you insert a distinct branch between a panel and another Script Author object, you must define the distinct answer choice that branch represents.

Distinct branches must originate from a panel object only. The distinct branch is the only branch type with this restriction.

Note: Inserting a distinct branch requires a value to be defined for that branch, as described below. The source of the value is the set of answer choices defined in the data dictionary for a question in that panel. If omitted, the script will not be syntactically correct.

Use this procedure to define a distinct branch leading from the source panel to the destination object. Defining the distinct branch includes specifying the answer choice from the question data dictionary that invokes that distinct branch path when selected at runtime.

Prerequisites

- Two objects (a panel and any other object) must exist on the canvas in order to connect them using the distinct branch.
- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

- Enable distinct branching for the source panel. In the Answer Entry Dialog window for any panel using distinct branching, select **Default for Distinct Branching.** You can do this when defining the question control, or later.
- Access the Data Dictionary for the question using distinct branching in a panel, and establish answer choices.

  Answer choice definitions include the answer choice *value* (passed by the application when selected at runtime), and the *display value* (the actual text shown to the end user to select). Answer choices are referred to in the UI as *lookup values*. When defining distinct branches, the full list of answer choices defined for that question (less those choices already assigned to other distinct branches from that question in the panel) is the source of selections (appearing in the Value pane of the Current Unused Answers list).
- Insert a destination object.
- Insert a distinct branch from the source panel to any destination object.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a distinct branch.

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

   The Value pane appears. This contains two lists.

   - The Current Unused Answers list contains the display values for answer choices for this question in the source panel that have not yet been assigned to a distinct branch.
   - The Selected Answer list contains the display value for the answer choice (if any) that is assigned to the selected distinct branch.
4. In the Value pane, do the following:

   a. In the Current Unused Answers list, click the answer choice that you want to trigger the branch.

   b. Click the double right-arrow button to move the answer to the Selected Answer list.

   c. Confirm that the correct answer choice appears in the Selected Answer list.

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

See Also

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Default Branch
- Defining a Conditional Branch
- Defining an Indeterminate Branch
- Defining the Branch Name
- Reordering Branches

References

- Working with Branches on the Canvas

4.11.4 Defining a Conditional Branch

The conditional branch leads from the source object to a destination object on the canvas, and navigates the script to its destination object when the conditional expression associated with that branch evaluates as true at runtime.

Use this procedure to define the conditional branch, including specifying a Script Author Java command as the expression evaluated by the branch at runtime.

Note: The conditional branch requires a conditional expression to be associated with that branch as a Java command, as described below. If the command is omitted, the script will not be syntactically correct.

Note: During a script syntax check, there is no validation to determine whether the custom code referenced by the Java command exists. Nonetheless, the referenced custom code must be available in runtime in order for the script to execute. Custom code must be available to the base Java classes that provide script runtime functionality. 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*.

Prerequisites

- Two objects must exist on the canvas in order to connect them using the conditional branch.
- Write the code for the condition (typically a Java method).
- Insert a conditional branch.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a conditional branch.

   The Properties window appears.
3. In the Conditional tree, select **Conditions**.

   The Conditions pane appears.
4. In the Conditions pane, click **Edit**.

   The Command window appears.
5. Define the condition as a Java command.

   Reference a Java command available to the Scripting Engine at runtime that returns a value of **true** or **false**.

   For more information, see Defining a Java Command.
6. Click **OK** to exit the Command window.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

References

- Working with Branches on the Canvas
- Defining a Java Command

See Also

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Default Branch
- Defining a Distinct Branch
- Defining an Indeterminate Branch
- Defining the Branch Name
- Reordering Branches

4.11.5 Defining an Indeterminate Branch

The indeterminate branch is used when the destination of the script is uncertain until information is gathered in a script session and evaluated. The information evaluated can include:

- 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). This command is specified as an expression associated with the branch.

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. The return value of the 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.

The command must return a value that specifies an object name or shortcut name in the script. The destination object must be either the *panel name* of a sibling panel (a panel on the same graph as the indeterminate branch), or the *shortcut name* of a group at any level of the script.

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.

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.

Note: The indeterminate branch requires an expression to be associated with that branch as a Java command, as described below. If the command is omitted, the script will not be syntactically correct.

Note: During a script syntax check, there is no validation to determine whether the custom code referenced by the Java command exists. Nonetheless, the referenced custom code must be available in runtime in order for the script to execute. Custom code must be available to the base Java classes that provide script runtime functionality. 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*.

Use this procedure to define the indeterminate branch, including specifying a Script Author Java command as the expression evaluated by the branch at runtime.

Prerequisites

- A source object (a panel, group or block object) must exist on the canvas in order to use the indeterminate branch.
- If jumping to a destination object on a different graph (whether higher or lower in the script), you must jump to a group, using the shortcut name provided for that group. If you want to jump to a panel or block, use a group containing a shortcut that you reference, and include the intended panel or block as the first configurable object within the destination group.

- Write the code for the expression (typically a Java method).
- Insert an indeterminate branch.
- Associate a Java command with the expression property of the indeterminate branch.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click an indeterminate branch.

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

   The Command window appears.
5. Define the indeterminate expression as a command.

   The command must return the name of:

   a. A "sibling" object (a panel, block or group on the same graph as the source object)

   b. The Shortcut name associated with a group on any hierarchical level of the script

   For more information, see Defining a Java Command.
6. Click **OK** to exit the Command window.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Default Branch
- Defining a Distinct Branch
- Defining a Conditional Branch
- Defining the Branch Name
- Reordering Branches

References

- Working with Branches on the Canvas

4.11.6 Defining the Branch Name

Use this procedure to define the name of the branch. Except for default branches, the branch name appears with the branch on the script canvas. Default branches, even if provided a customized name, do not display a branch label on the canvas.

Prerequisites

Insert a branch onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a branch.

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

See Also

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Default Branch
- Defining a Distinct Branch
- Defining a Conditional Branch
- Defining an Indeterminate Branch
- Reordering Branches

References

- Working with Branches on the Canvas

4.11.7 Reordering Branches

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

Prerequisites

- Insert a configurable object (panel, group or block) onto the canvas.
- Insert two or more branches onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a configurable object on the canvas.

   The Properties window appears.
3. In the Panel tree, select **Branches**.
4. In the Branches pane, select a branch.

   The order in which branches are numbered (from top to bottom in the Branches pane) indicates the branch order, as does the number in brackets following the branch name.

   The lowest branch order (first branch included in the list) is evaluated first, the highest branch order is evaluated last.
5. If you want a branch to be evaluated at runtime *earlier* than other outgoing branches for this object, then select the second or subsequent branch from the Branches pane and click **Move Up**. Do this as many times as required until the selected branch appears in the correct order in the list.

   The priority for the evaluation of the selected branch is increased or moved up by one increment for each time you Move Up (until that branch is listed first in the list).
6. If you want a branch to be evaluated at runtime *later* than other outgoing branches for this object, then select any branch other than the last in the list and click **Move Down**. Do this as many times as required until the selected branch appears in the correct order in the list.

   The priority for the evaluation of the selected branch is lowered or moved down by one increment for each time you Move Down (until that branch is listed last in the list).
7. When the branch order is increased or lowered as appropriate, then click **Apply**.

When you apply the change, the number in brackets is updated, changing by one level for each time you clicked it.

The branch order [0] indicates the highest priority of evaluation, and indicates that the selected branch is listed first.

The branch listed last has the highest number in brackets, indicating that it is processed in that order.

Note: If you move the branch up or down, the branch order indicated within brackets does not change until you click **Apply**.

---

8. Click **OK** to save your work and exit the Properties window.

References

- Working with Branches on the Canvas
- Inserting a Panel
- Inserting a Group
- Inserting a Block

See Also

- Setting Properties Window to Pop Up at Branch Creation
- Defining a Default Branch
- Defining a Distinct Branch
- Defining a Conditional Branch
- Defining an Indeterminate Branch
- Defining the Branch Name

4.12 Defining Database Connections

When a script administrator (such as a script developer) logs into Script Author, authentication information from the user's Oracle Applications session is used automatically.

When creating a script, the connection to the database for that database instance is maintained during an active session. This is referred to as the *login connection*.

When performing actions that directly utilize the database, such as deploying scripts, storing or accessing stored commands, or performing PL/SQL actions, the login connection to the database is used automatically.

Database connections can also be defined in a script to other database tables or views. Connection information for database connections other than the login connection is now maintained in Script Author's Connection List (**Script** > **Connection List**). The Connection List window displays all connections defined for the script.

Each script has its own set of defined connections. If a script with one or more database connections defined (for example, Script A) is closed, and a new script (Script B) is created, there are no connections defined for the new script. However, when Script A is *imported* into script B, the connections from Script A will also be copied into Script B, so that in Script B, connections from both scripts are available in the Connection List dialog.

Use this procedure to enter, modify, or test database connection information for a script.

1. From a graphical script, select **Script** > **Connection List**.

   The Connection List window appears.

2. If you want to add database connection information, perform the following:

   a. Click **Add.**

      The Connection window appears.

   b. In the Connection window, select the **Connection** tab.

   c. In the Connection tab, in the Connection Name field, type a meaningful name.

      This name identifies this connection and will subsequently appear in the Connection list for this script.

   d. In the Host Name field, type the appropriate database host, including domain if appropriate.

   e. In the Port Number field, type the port number of the database instance to which you want to connect.

   f. In the SID field, type the SID of the database instance to which you want to connect.

   g. In the User Name field, type the username of an Oracle Applications user with apps level access to the appropriate database instance.

For example, type apps.

h. In the Password field, type the appropriate password for this user.

i. To save your connection information, click **OK**.

The Connection window closes. The Connection List window appears. The connection you defined appears in the list.

3. If you want to edit information for an existing database connection, perform the following:

a. In the Connection List window, in the Connection List area, click **Edit**.

A warning appears, indicating that any objects using this connection will be affected by the change, and asking if you want to edit the connection.

* To cancel this request and return to the Connection List window, click **No**.

* To proceed, click **Yes**.

The Connection window appears.

b. In the connection window, edit information in any field as appropriate.

c. To save your connection information, click **OK**.

4. If you want to test a defined database connection, then do the following:

a. From the Script menu, select **Connection List**.

The Connection List window appears. Each configured connection appears in the list.

b. Select the connection you want to test and click **Edit**.

A warning appears, indicating that any objects using this connection will be affected by the change, and asking if you want to edit the connection.

c. In the warning window, click **Yes**.

The Connection window appears.

d. Select the Test tab.

e. Click the **Test Connection** button.

View the status of the connection in the Status area.

f. To exit, click **OK** or **Cancel**.

The Connection window closes. The Connection List window appears.

g. To exit the connection list, click **Close.**

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.13 Controlling Script Flow

Script flow is controlled by several aspects. The first and most obvious method is by defining. However, script flow is also controlled by other aspects. The following list includes Script Author functions that control script flow.

Note: Each of the topics below are intermediary topics, and are cross-referenced below only. The content for each is included in its own relevant category.

- Defining Branches
- Defining Panel Question UI Controls
- Defining Actions
- Defining Blocks
- Defining Commands
- Defining Shortcuts

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands

- Reusing Commands
- Deploying the Script

4.14 Using the Panel Layout Editor

Each panel in a script displays at least one question user interface control 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. Question controls defined for a panel automatically appear in the panel layout. Additionally, you can create, modify or add optional panel content in a graphical script using the *panel layout editor.*

You can perform the following tasks:

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor
- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels

- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.14.1 Understanding the Panel Layout Editor

This section includes the following topics:

- Intended Uses and Limitations of the Panel Layout Editor
- Customizations Involve Importing and Exporting Panel HTML
- Panel HTML Contains Script-Specific Code
- What Displays in the Panel Layout Editor
- What Is Not Supported in Panel Layout HTML

See Also

- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

4.14.1.1 Intended Uses and Limitations of the Panel Layout Editor

The panel layout editor is a feature of Script Author for defining and refining the information contained in a panel in a graphical script. Using the panel layout editor, script developers can quickly define simple HTML layouts for a panel without writing HTML code.

The panel layout editor includes one-click formatting of text into two distinct styles. *Instructional text* indicates to the script end user specific instructions. *Spoken text* indicates text 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.

Other formatting can be applied to panel layout content as well, using the full range of formatting tools discussed in this section. However, only spoken text and instructional text styles are available for single-click uniform panel design.

You can also include some complex functions using the panel layout editor, such as including embedded values in a panel. 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 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, perform one-click text formatting, create ordered or unordered lists, or select from a variety of typeface styles, sizes, and formatting to apply to panel layout items.

Not a Replacement for Full-Featured HTML Editors

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.

See Also

- Customizations Involve Importing and Exporting Panel HTML
- Panel HTML Contains Script-Specific Code
- What Displays in the Panel Layout Editor

- What Is Not Supported in Panel Layout HTML

4.14.1.2 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*, release 11i.

See Also

- Intended Uses and Limitations of the Panel Layout Editor
- Panel HTML Contains Script-Specific Code
- What Displays in the Panel Layout Editor
- What Is Not Supported in Panel Layout HTML

4.14.1.3 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*, release 11i.

See Also

- Intended Uses and Limitations of the Panel Layout Editor
- Customizations Involve Importing and Exporting Panel HTML
- What Displays in the Panel Layout Editor
- What Is Not Supported in Panel Layout HTML

4.14.1.4 What Displays in the Panel Layout Editor

Question UI 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.

Label Values and the Panel Layout Editor

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. The behavior of this panel element is worthy of note.

After the value in the Label for Reporting field is confirmed (by clicking OK in the Answer Entry Dialog window), you can still change the value (by opening this window a subsequent time, for the same question in the panel). Modifying this

value (or adding one where initially no value was entered) results in updating the label value associated with its panel question (when used for reporting purposes), and changes the value appearing in the Answer Entry Dialog window. It also updates the string associated with the panel question (as the question label) in the database.

However, the only time label content is generated in panel HTML is when the question control is initially saved. Thus, 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.

See Also

- Intended Uses and Limitations of the Panel Layout Editor
- Customizations Involve Importing and Exporting Panel HTML
- Panel HTML Contains Script-Specific Code
- What Is Not Supported in Panel Layout HTML

4.14.1.5 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 stand-alone 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.

See Also

- Intended Uses and Limitations of the Panel Layout Editor
- Customizations Involve Importing and Exporting Panel HTML
- Panel HTML Contains Script-Specific Code
- What Displays in the Panel Layout Editor

4.14.2 Opening the Panel Layout Editor

Use this procedure to open the panel layout editor from a graphical script.

Prerequisites

- Insert a panel onto the canvas.
- Optionally, define the questions required for this panel.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, select a panel.
3. Open the Panel Layout Editor using one of the following methods:
   - On your keyboard, select **Ctrl + Shift + E**.
   - From the menu, choose **Tools > Panel Layout Editor**.
   - From the toolbar, click the **Panel Layout** icon.

   The panel layout editor appears.

See Also

- Understanding the Panel Layout Editor

- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

4.14.3 Entering and Formatting Panel Text

Use this procedure to define the text that displays in the script panel at runtime, and to format panel text and any panel labels defined in the Answer Entry dialog window.

Oracle Corporation recommends defining panel questions (and values in the Label for Reporting field, if applicable) prior to formatting panel text and layout.

Prerequisites

- Insert a panel onto the canvas.
- Optionally, define the questions required for this panel.

Steps

1. Open the panel layout editor for a selected panel.

   The panel layout editor appears.

2. Type the panel text.
3. Optionally, select text and then format it using one of the following methods:
   - Select a format from the **Font** menu.
   - To set one or more selected paragraphs into an ordered list, from the toolbar, select **Insert List**.
   - 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.

4. If you want to insert a bulleted or a cardinally ordered list, then do the following:
   a. Without selecting text first, place the cursor at the appropriate insertion point in the panel.
   b. 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.
   c. Type or paste text into the list as appropriate. To add new items to the list, on your keyboard, press **Return**.

      A new bullet, or the next numeral in ordinal sequence, respectively, appears on the next line, ready for you to type more text.
5. If you want to save the panel text, then choose **File > Save** from the menu.
6. 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: 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 (*.*)**.

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

See Also

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

4.14.4 Inserting a Hypertext Link

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.

Prerequisites

Insert a panel onto the canvas.

Steps

1. Open the panel layout editor for a selected panel.

   The panel layout editor appears.

2. If required, type the text you want to display as the hypertext link.
3. Select the appropriate text and then click **Insert Link** in the toolbar.

   The selected text is blue and underlined.

4. With the cursor in the text or with the text selected, click **Modify Properties** in the toolbar.

   The Hypertext Link window appears.

5. In the Hypertext Link window, do one of the following:
   - If you want to define the hypertext link using a URL (Uniform Resource Locator, or web address), then, 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.
6. Click **OK** to exit the Hypertext Link window.
7. 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: Exporting the text to an HTML file does not save the text to the panel.

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

See Also

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor

- Entering and Formatting Panel Text
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

4.14.5 Inserting an Embedded Value

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

Prerequisites

Insert a panel onto the canvas.

Steps

1. Open the panel layout editor for a selected panel.

   The panel layout editor appears.

2. Type placeholder text for the embedded value.

   For example, if creating an embedded value to display the blackboard command of a customer's first name, type **first_name**.

3. Select the text placeholder and then click **Insert Embedded Value** in the toolbar.

   The selected text is marked as an embedded value, providing a unique Oracle Scripting embedded value ID (iesuid). When viewed in the panel layout editor without any text selected, the background of embedded value is yellow.

   You can now modify properties for the embedded value.

4. With the cursor in the text or with the embedded value text selected, from the toolbar, click **Modify Properties**.

   The Command window appears.

5. Define a command that returns a value to be displayed in the script panel at runtime. (See Defining Commands.)

   For example, to provide the known value of a blackboard key "first_name" to populate as the embedded value in the panel at runtime, do the following:

   - From the Command Type menu, select Blackboard Command.

- In the Name field, type a command name. The value you enter here will be visible from the panel layout editor. For example, to define a name for this command that defines it as a blackboard command, type bb_first_name.
- In the command field, enter the blackboard key (for example, first_name). The value you enter here, if known at runtime, will display in the panel dynamically at runtime.
- Click **OK**.

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

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

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

See Also

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor
- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

4.14.6 Inserting an Image

The panel layout editor includes an Insert Image button in the toolbar. This allows the script developer to locate (from the local file system or network) an existing GIF or JPG image, and place the specified image in the panel. There are two primary caveats with this process:

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 that 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.

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

Prerequisites

- Insert a panel onto the canvas.
- Identify a graphic image to import.
- Load the graphic to a Web server location accessible to the Scripting Engine at runtime.

Steps

1. Open the panel layout editor for a selected panel.

   The panel layout editor appears.
2. Position the insertion point where you want to insert the picture.
3. In the toolbar, click **Insert Image**.

   The Open window appears.
4. From your local file system or network, locate the file that contains the image you want to insert.

   Specify a graphics interchange format (.GIF) or joint photographics expert group format (JPEG) file.

5. Click the file and 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.

6. Optionally, make any layout changes necessitated by placement of the graphic into your panel layout.
7. Save the panel text by choosing **File > Save** from the menu.
8. Export the panel contents to an HTML file (see Exporting Panel Text to an HTML File).
9. 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">`.

10. 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">
    ```

11. Make any other required modifications to the panel layout HTML file, and save it in HTML format.
12. Import the file into the original panel, replacing the old panel layout contents with the modified HTML. (For more information, see Importing an HTML File into the Panel Layout Editor.)

    > **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 (*.*)**.

13. If you want to save the modified panel content, then choose **File > Save.**

See Also

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor
- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

4.14.7 Exporting Panel Text to an HTML File

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**.

Prerequisites

Insert a panel onto the canvas.

Steps

1. Open the panel layout editor for a selected panel.

   The panel layout editor appears.

2. From the **File** menu, select **Export As**.

   The Save window appears.

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.

4. Click **Save**.

   The panel contents are exported using the designated file name and location.

5. Optionally, to save the current panel contents, select **File > Save.**

See Also

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor
- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File

4.14.8 Importing an HTML File 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.
>
> 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.
>
> 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.

Prerequisites

- Insert a panel onto the canvas.
- Identify an HTML file to import into a panel.

Steps

1. Open the panel layout editor for a selected panel.

The panel layout editor appears.

2. Choose **File > Import**.

   The Open window appears.

3. Locate the file that you want to import.

   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 (*.*)**.

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

   The HTML appears in the panel layout editor.

5. Optionally, if you want data dictionary properties associated with any questions that are new to the panel, enter them as appropriate.

6. If you want to save the imported panel content, then choose **File > Save.**

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.

See Also

- Understanding the Panel Layout Editor
- Opening the Panel Layout Editor
- Entering and Formatting Panel Text
- Inserting a Hypertext Link
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File

4.15 Defining Panel Question UI Controls

Question UI controls are the user interface controls that display in a script at runtime. Every panel requires at least one question or node to be defined. Each node may contain a single user interface type, and may have text associated with the question UI control. The end user (interaction center agent, survey respondent or Web script user) must interact with a question UI control for each panel, whether the panel contains a single node or multiple nodes. If a question includes validation, that panel at runtime requires the end user to interact with that question UI control following the parameters dictated by the code behind the validation command.

You can perform the following tasks:

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel
- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

References

- Inserting a Panel
- Defining Panel Properties
- Defining Panel Text and Layouts

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas

- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.15.1 Defining a Text Field Question UI Control in a Panel

Use this procedure to define a text field question UI control in a script panel. A text field is a text entry field that displays a single row for text entry. The text field is generally intended for use to capture text answers of one line or less in runtime. Beyond the space provided, the end user can continue to enter characters with no limit. There is no restriction to the content allowed in a text field unless validation is associated with this node in the data dictionary for the text field question control.

Prerequisites

Insert a panel onto the canvas.

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 **Answers**.

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

   The Answer Entry Dialog window appears.

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.

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

   Note: The answer name must be unique.

7. From the UI Type list, select **Text Field**.

8. Optionally, in the Label for Reporting field, type a label.

   This label will appear at runtime to the left of the text field in the panel. If used with the survey component, this field will provide a label for reporting purposes.

9. Click **OK** to exit the Answer Entry Dialog window.

   The answer appears in the Answer pane in the Properties window.

10. If you want to define another text field in the same script panel, then repeat steps 4 through 9.

11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel

- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.2 Defining a Text Area Question UI Control in a Panel

Use this procedure to define a text area question UI control in a script panel. A text area is a text entry field that displays multiple rows for text entry. The text area is generally intended for use to capture text answers longer than one line in runtime. Beyond the space provided, the end user can continue to enter characters with no limit. There is no restriction to the content allowed in a text area unless validation is associated with this node in the data dictionary for the text field question control.

Prerequisites

Insert a panel onto the canvas.

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

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.

7. From the UI Type list, select **Text Area**.
8. Optionally, in the Label for Reporting field, type a label.

   This label will appear at runtime to the left of the text area you define for this question UI control. If used with the survey component, this field will display as a label for reporting purposes.
9. Click **OK** to exit the Answer Entry Dialog window.

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

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel
- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.3 Defining a Radio Button Group Question UI Control in a Panel

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

Prerequisites

Insert a panel onto the canvas.

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

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, choose **Radio Button**.
8. Optionally, in the Label for Reporting field, type a label.

   This label will appear at runtime to the left of the answer choices you define for this question UI control. If used with the survey component, this field will serve as a label for reporting purposes.
9. Click **Edit Data Dictionary**.

   The Edit Data Dictionary window appears.
10. Click the Lookups tab.

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

    The Lookup Entry window appears.
11. In the Display Value field, type the text that appears next to the radio button in the script panel at runtime.
12. In the Value field, type the value that is stored in the Oracle Scripting database schema when the radio button is selected at runtime.
13. If you want to add another radio button to the group, then repeat steps 10 through 12.

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

    The lookup entry values you provided as choices for this answer in runtime appear in the Specify Lookups window.

15. Click **OK** to exit the Edit Data Dictionary window.

16. Click **OK** to exit the Answer Entry Dialog window.

17. If you want to define another radio button group in the same script panel, then repeat steps 4 through 15.

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

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel
- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.4 Defining a Check Box Group Question UI Control in a Panel

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

Prerequisites

Insert a panel onto the canvas.

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 **Answers**.

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

   The Answer Entry Dialog window appears.

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.

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

   Note: The answer name must be unique.

7. From the UI Type list, choose **Check Box**.

8. Optionally, in the Label for Reporting field, type a label.

   This label will appear at runtime to the left of the answer choices you define for this question UI control. If used with the survey component, this field will serve as a label for reporting purposes.

9. Click **OK** to exit the Answer Entry Dialog window.

   The answer appears in the Answer pane in the Properties window.

10. If you want to define another check box in the same script panel, then repeat steps 4 through 9.

11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel

- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.5 Defining a Button Question UI Control in a Panel

Use this procedure to define a buttons in a script panel.

> **Note:** The button UI type question UI control must only be used in panels that contain *a single node or question.* Use of buttons for more than one node in a single panel *is not supported.*

Prerequisites

- Insert a panel onto the canvas.
- Ensure only one question is required in this panel.

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

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

   > **Note:** If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.

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

Note: The answer name must be unique.

7. From the UI Type list, choose **Button**.
8. Click **Edit Data Dictionary**.

   The Edit Data Dictionary window appears.
9. In the Lookups tab, select **Specify lookups** and then click **Add**.

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

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel
- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.6 Defining a Drop Down List Question UI Control in a Panel

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

Prerequisites

Insert a panel onto the canvas.

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

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, choose **Drop Down**.
8. Optionally, in the Label for Reporting field, type a label.

   This label will appear at runtime to the left of the answer choices you define for this question UI control. If used with the survey component, this field will serve as a label for reporting purposes.
9. Click **Edit Data Dictionary**.

   The Edit Data Dictionary window appears.
10. In the Lookups tab, select **Specify lookups** and then click **Add**.

    The Lookup Entry window appears.

11. In the Display Value field, type the text that appears in the drop down list in the script panel at runtime.
12. In the Value field, type the value that is stored in the Oracle Scripting database schema when the drop down is selected at runtime.
13. If you want to add another item to the drop down list, then repeat steps 10 through 13.
14. Click **OK** to exit the Lookup Entry window.

    The lookup entry values you provided as choices for this answer in runtime appear in the Specify Lookups window.
15. Click **OK** to exit the Edit Data Dictionary window.
16. Click **OK** to exit the Answer Entry Dialog window.

    The answer appears in the Answer pane in the Properties window.
17. If you want to define another drop down list in the same script panel, then repeat steps 4 through 15.
18. If you want to define a validation command for this password field, follow the procedure in Adding Validation to a Question.
19. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.7 Defining a Password Question UI Control in a Panel

Use this procedure to define a password question UI control in a script panel. A password field is a single-row text entry field that displays the input as asterisks at runtime. Any password functionality such as validation against a valid range of values must be developed independently and associated with the password question UI control using the Validation command in the data dictionary. This feature is not automatically included in this question UI control type.

Prerequisites

Insert a panel onto the canvas.

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

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, select **Password**.
8. Optionally, in the Label for Reporting field, type a label.

   For example, type PASSWORD.

This label will appear at runtime to the left of the password field. If used with the survey component, this field will serve as a label for reporting purposes.

9. Click **OK** to exit the Answer Entry Dialog window.

   The answer appears in the Answer pane in the Properties window.

10. If you want to define another password in the same script panel, then repeat steps 4 through 9.

11. If you want to add validation to this password, see Adding Validation to a Question.

12. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.8 Defining a Checkbox Group Question UI Control in a Panel

Use this procedure to define a checkbox group question UI control in a script panel. A password field is a single-row text entry field that displays the input as asterisks at runtime. Any password functionality such as validation against a valid range of values must be developed independently and associated with the password question UI control using the Validation command in the data dictionary. This feature is not automatically included in this question UI control type.

Prerequisites

Insert a panel onto the canvas.

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

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, select **Checkbox Group**.
8. Optionally, in the Label for Reporting field, type a label.

   This label will appear at runtime to the left of the answer choices you define for this question UI control. If used with the survey component, this field will serve as a label for reporting purposes.
9. Click **Edit Data Dictionary**.

   The Edit Data Dictionary window appears.
10. Click the Lookups tab.

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

    The Lookup Entry window appears.
11. In the Display Value field, type the text you want to appear for this question UI control.

12. In the Value field, type the value you want to be passed to Oracle Scripting when this answer choice is selected at runtime.
13. Click OK.

    The Lookup Entry window closes. The answer choice you have just defined appears in the Specify Lookups area, in the format **display value:value.**
14. For each additional lookup value or answer choice you want to populate in the checkbox group, repeat steps 10 through 13.
15. Click **OK** to exit the Edit Data Dictionary window.
16. Click **OK** to exit the Answer Entry Dialog window.

    The question you have defined appears in the Answer pane in the Properties window.
17. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel
- Defining a Password Question UI Control in a Panel
- Defining a Multi-Select List Box Question UI Control in a Panel

4.15.9 Defining a Multi-Select List Box Question UI Control in a Panel

Use this procedure to define a multi-select list box question UI control in a script panel. A multi-select list box question UI control is the same as the drop down list question UI control, except that more than one of the lookup values may be selected at runtime. As in the past, clicking on any lookup value or option in the list will highlight that single item. Now, holding down the Control key (on a Microsoft Windows platform) or the Option key (on a Macintosh OS platform) will allow any two or more lookup values or list options to remain selected. Clicking the Shift key will allow the user to select any 2 or more consecutive answers.

Like the checkbox group question UI control, the multi-select list box question UI control supports multiple answer selection by storing a vector of answers instead of a single answer. This feature is introduced in Script Author release 11.5.7 RUP 2. Scripts using multiple-answer UI controls should be edited only in Script Author 1.6.1.00 or later and executed in a Scripting Engine that has been upgraded to 11.5.7 RUP 2 or later.

This answer UI control type requires one or more values to be defined as lookups in the data dictionary to be syntactically correct. It requires two or more values to be defined as lookups in order to be functionally distinct from the standard list box at runtime.

Prerequisites

Insert a panel onto the canvas.

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

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

   Note: If using distinct branching from this panel, at least one question *must* be designated as the default answer for distinct branching. Selecting this checkbox is optional if distinct branching is not required.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, scroll down if necessary and select **Multi-Select List Box**.

8. Optionally, in the Label for Reporting field, type a label.
9. This label will appear at runtime to the left of the answer choices you define for this question UI control. If used with the survey component, this field will serve as a Click **Edit Data Dictionary**.

   The Edit Data Dictionary window appears.
10. Click the Lookups tab.

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

    The Lookup Entry window appears.
11. In the Display Value field, type the text you want to appear for this question UI control.
12. In the Value field, type the value you want to be passed to Oracle Scripting when this answer choice is selected at runtime.
13. Click OK.

    The Lookup Entry window closes. The answer choice you have just defined appears in the Specify Lookups area, in the format **display value:value.**
14. For each additional lookup value or answer choice you want to populate in the drop-down list, repeat steps 10 through 13.
15. Click **OK** to exit the Edit Data Dictionary window.
16. Click **OK** to exit the Answer Entry Dialog window.

    The question you have defined appears in the Answer pane in the Properties window.
17. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining a Text Field Question UI Control in a Panel
- Defining a Text Area Question UI Control in a Panell
- Defining a Radio Button Group Question UI Control in a Panel
- Defining a Check Box Group Question UI Control in a Panel
- Defining a Button Question UI Control in a Panel
- Defining a Drop Down List Question UI Control in a Panel

- Defining a Password Question UI Control in a Panel
- Defining a Checkbox Group Question UI Control in a Panel

4.16 Working with Questions

After defining a question for a panel, you can perform the following tasks:

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow

- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Defining Actions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.16.1 Accessing the Data Dictionary for a Question

In a graphical script, the data dictionary provides a method to associate properties specific to a question in a panel. These are properties which do not appear in panel HTML, and include commands, table information, and answer choices (lookups) associated with the question.

Use this procedure to access the data dictionary for a specific question in a panel.

Prerequisites

- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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 **Answers**.

   The Answers pane appears.
4. From the Answers pane, select the appropriate question and click **Edit**.

   The Answer Entry Dialog window appears.
5. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog appears.
6. If you want to associate answer validation, a default command, or a data source with this question, then from the General tab, in the relevant field, click **Edit** and associate the command as appropriate.

7. If you want to associate table information for a database table into which responses are to be stored, then in the Table info tab, enter information as appropriate:

   a. In the Table Data area, in the Table field, type the name of the table into which the answer is inserted.

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

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

8. If you want to populate answer choices into a question, then from the Lookups tab, in the Lookups area, do the following:

   a. To obtain answer choices from a specific table, select **Table lookup** and define the lookup table name.

      This option requires additional table information to be specified in the Table info tab of the question's data dictionary.

   b. 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.

   c. 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.

   d. To hard-code specific values as answer choices, select Specify lookups and then click Add for each set of lookup values and the display values for that answer choice.

9. From the Edit Data Dictionary window, click **Ok** to save your data dictionary work.

   The Edit Data Dictionary window closes. The Answer Entry Dialog window appears.

10. From the Answer Entry Dialog window, click **Ok** to save your answer properties.

The Answer Entry Dialog window closes. The Panel Properties window appears.

11. From the Panel Properties window, click **Ok** to save your work and close the window, or **Apply** to save your work and continue adding or modifying panel properties.

See Also

- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

4.16.2 Adding Validation to a Question

Question validation establishes and enforces business rules for a designated question control (node) in a panel. When the script end user provides a response to that 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.

To add validation to a question in Script Author, access the data dictionary for the question. In the General tab, define a Java command using the Validation option.

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.

Using Custom Code

You can define custom validation for a question 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)

Use this procedure to define a validation command to be applied at runtime against an answer provided into the panel question UI control.

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.
- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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 **Answers**.
4. From the Panel Properties dialog, select the appropriate question and click **Edit**.

   The Answer Entry Dialog window appears.
5. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog appears.
6. In the General tab, click **Edit** on the Validation control

   The Command window appears.
7. Define a Java command for the validation action. (See Defining Commands.) Ensure you include any required parameters.
8. Click **OK** to exit the Command window.
9. Click **OK** to exit the Data Dictionary window.
10. Click **OK** to exit the Answer Entry Dialog window.
11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

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 question for which you want the validation routine to apply. In the Parameter window, in the Name field, type the node or question name.
  - The label parameter defines the label of the question control 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

See Also

- Accessing the Data Dictionary for a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

4.16.3 Defining Data Constraints for an Question UI Control

Use this procedure to define data constraints for a question UI control.

Prerequisites

- Identify the data constraints applicable to this answer
- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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 **Answers**.
4. From the Panel Properties dialog, select the question and click **Edit**.

   The Answer Entry Dialog window appears.

5. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog appears.

6. In the General tab, define the data constraints for the question UI control.
7. Click **OK** to exit the Data Dictionary window.
8. Click **OK** to exit the Answer Entry Dialog window.
9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

4.16.4 Reordering Answer Options

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

Prerequisites

- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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 **Answers**.
4. From the Panel Properties dialog, select the question and click **Edit**.

The Answer Entry Dialog window appears.

5. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog appears.

6. In the Lookups tab, select a value.
7. If you want to move the option up, then click **Move Up**.
8. If you want to move the option down, then click **Move Down**.
9. Click **OK** to exit the Edit Data Dictionary window.
10. Click **OK** to exit the Answer Entry Dialog window.
11. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

4.16.5 Reordering Question UI Controls

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

Prerequisites

- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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

See Also

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

4.16.6 Deleting Answer Options from a Question UI Control

Use this procedure to delete an options from a question UI control.

Prerequisites

- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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 **Answers**.
4. From the Panel Properties dialog, select the question and click **Edit**.

The Answer Entry Dialog window appears.

5. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog appears.

6. In the Lookups tab, select a value and then click **Remove**.
7. Click **OK** to exit the Edit Data Dictionary window.
8. Click **OK** to exit the Answer Entry Dialog window.
9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Question UI Controls from a Script Panel
- Substituting a Java Bean for an Answer

4.16.7 Deleting Question UI Controls from a Script Panel

Use this procedure to delete a question UI control from a script panel.

Prerequisites

- Insert a panel onto the canvas.
- Define a question UI control for the script panel.

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 **Answers**.

4. In the Answers pane, select an answer and then click **Remove**.
5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Substituting a Java Bean for an Answer No Longer Supported

4.16.8 Substituting a Java Bean for an Answer

Substituting a Java bean for an answer (or panel node) is no longer supported in Oracle Scripting as of release 11.5.6 and subsequent releases, due to the new WYSIWYG editing feature of Script Author.

See Also

- Accessing the Data Dictionary for a Question
- Adding Validation to a Question
- Defining Data Constraints for an Question UI Control
- Reordering Answer Options
- Reordering Question UI Controls
- Deleting Answer Options from a Question UI Control
- Deleting Question UI Controls from a Script Panel

4.17 Defining Actions

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.

If using an API, then adding 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 provides the added advantage to script developers of a clear visual indicator that an action is referenced.

You can perform the following tasks:

- Defining Script Pre- and Post-Actions
- Defining Panel Pre- and Post-Actions
- Defining Group Pre- and Post-Actions
- Defining Block Pre- and Post- Actions
- Defining Block API Actions
- Defining Branch Actions

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor

- Defining Panel Question UI Controls
- Working with Questions
- Defining Commands
- Reusing Commands
- Deploying the Script

4.17.1 Defining Script Pre- and Post-Actions

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

Prerequisites

Create or open a script.

Steps

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

   The Properties window appears.

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

   The Command window appears.

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

See Also

- Defining Panel Pre- and Post-Actions
- Defining Group Pre- and Post-Actions
- Defining Block Pre- and Post- Actions
- Defining Block API Actions
- Defining Branch Actions

4.17.2 Defining Panel Pre- and Post-Actions

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

Prerequisites

Insert a panel onto the canvas.

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

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

See Also

- Defining Script Pre- and Post-Actions
- Defining Group Pre- and Post-Actions
- Defining Block Pre- and Post- Actions
- Defining Block API Actions
- Defining Branch Actions

4.17.3 Defining Group Pre- and Post-Actions

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

Prerequisites

Insert a group onto the canvas

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

   The Command window appears.

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

See Also

- Defining Script Pre- and Post-Actions
- Defining Panel Pre- and Post-Actions
- Defining Block Pre- and Post- Actions
- Defining Branch Actions

4.17.4 Defining Block Pre- and Post- Actions

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

Prerequisites

Insert a block onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a block.

   The Properties window appears.

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

   The Command window appears.

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

See Also

- Defining Script Pre- and Post-Actions
- Defining Panel Pre- and Post-Actions
- Defining Group Pre- and Post-Actions
- Defining Block API Actions
- Defining Branch Actions

4.17.5 Defining Block API Actions

Use this procedure to define an API block to reference an action to take place when the flow of the script reaches the block. This action will occur after any block pre-action and prior to any block post-action, and provides a visual indicator to script developers that a Script Author command exists at that point in the flow of the script.

Prerequisites

Insert a block onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a block.

   The Properties window appears.
3. In the Block tree, expand **Actions** and then select **Types.**
4. In the Types pane, select **API Block**.
5. In the Block tree, select **Object Dictionary.**
6. In the API Actions pane, click **Add**.
7. Define a command for the action. (See Defining Commands.)
8. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties window.

See Also

- Defining Script Pre- and Post-Actions
- Defining Panel Pre- and Post-Actions

- Defining Group Pre- and Post-Actions
- Defining Branch Actions

4.17.6 Defining Branch Actions

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

Prerequisites

Insert branches onto the canvas.

Steps

1. In the Object and Branch toolbar, click the Toggle Select Mode tool.
2. On the canvas, double-click a branch.

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

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

See Also

- Defining Script Pre- and Post-Actions
- Defining Panel Pre- and Post-Actions
- Defining Group Pre- and Post-Actions
- Defining Block Pre- and Post- Actions
- Defining Block API Actions

4.18 Defining Commands

You can perform the following tasks:

- Defining a Java Command
- Defining a PL/SQL Command

- Defining a Blackboard Command
- Defining a Forms Command
- Defining a Constant Command
- Defining a Delete Action

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Reusing Commands
- Deploying the Script

4.18.1 Defining a Java Command

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:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the command string as follows:

      ```
      <fully qualified class name>::<method name>
      ```

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

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

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

   e. Click **OK** to exit the Parameters window.
   f. If you want to add another parameter, then repeat steps a through e.
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*.

See Also

- Defining a PL/SQL Command
- Defining a Blackboard Command
- Defining a Forms Command
- Defining a Constant Command
- Defining a Delete Action

References

- *Oracle Scripting Implementation Guide*
- *Oracle Scripting Developer's Guide*

4.18.2 Defining a PL/SQL Command

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:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the name of a PL/SQL stored procedure or function in an Oracle database.
4. If you want to list parameters for the command object, then, in the Parameters area, do the following:
   a. Click **Add**.

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

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

   e. In the In/Out list, select the parameter type.
   f. Click **OK** to exit the Parameters window.
   g. If you want to add another parameter, then repeat steps a through f.
5. In the Connection area, specify the appropriate database connection for this PL/SQL command.

   If you want to connect to the same database instance used when logging into Oracle Applications, select **Use Login Connection**.

   If you want to connect to a different database instance, then 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 Defining Database Connections.

6. Click **OK** to exit the Command window.
7. Save your work.

See Also

- Defining a Java Command
- Defining a Blackboard Command
- Defining a Forms Command
- Defining a Constant Command
- Defining a Delete Action

4.18.3 Defining a Blackboard Command

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.

Prerequisites

None

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:
   a. In the Name field, type the name of the command.

b. 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.

See Also

- Defining a Java Command
- Defining a PL/SQL Command
- Defining a Forms Command
- Defining a Constant Command
- Defining a Delete Action

4.18.4 Defining a Forms Command

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:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the command string.

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

      The Parameters window appears.

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

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

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

      The Parameters window appears.

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

See Also

- Defining a Java Command
- Defining a PL/SQL Command
- Defining a Blackboard Command
- Defining a Constant Command
- Defining a Delete Action

4.18.5 Defining a Constant Command

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.

Prerequisites

None

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:
   a. Click **Edit**.

      The Parameters window appears.

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

See Also

- Defining a Java Command
- Defining a PL/SQL Command

- Defining a Blackboard Command
- Defining a Forms Command
- Defining a Delete Action

4.18.6 Defining a Delete Action

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.

See Also

- Defining a Java Command
- Defining a PL/SQL Command
- Defining a Blackboard Command
- Defining a Forms Command
- Defining a Constant Command

References

Oracle Scripting Developer's Guide

4.19 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 a Reusable Command
- Copying, Modifying, or Deleting a Reusable Command
- Importing a Reusable Command Into a Script

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions

- Defining Commands
- Deploying the Script

4.19.1 Defining a Reusable Command

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.

Prerequisites

None

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.

See Also

- Copying, Modifying, or Deleting a Reusable Command
- Importing a Reusable Command Into a Script

4.19.2 Copying, Modifying, or Deleting a Reusable Command

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:
   a. Locate the appropriate command in the command list.
   b. Select the command.
   c. Click **Copy**.

      Note: There may be a delay after this request as the command is retrieved from the database.

   d. 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:
   a. Locate the appropriate command in the command list.
   b. Select the command.

c. 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.

d. 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:

   a. Locate the appropriate command in the command list.

   b. Select the command.

   c. Click **Remove**.

      A warning dialog appears, with the message "Remove this command from the library?"

   d. 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.

See Also

- Defining a Reusable Command
- Importing a Reusable Command Into a Script

4.19.3 Importing a Reusable Command Into a Script

Once any 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:
   a. Locate the appropriate command in the command list.
   b. Select the command.
   c. 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.

See Also

- Defining a Reusable Command

- Copying, Modifying, or Deleting a Reusable Command

4.20 Deploying the Script

You can perform the following tasks:

- Checking Script Syntax
- Deploying a Script to the Database from Script Author
- Deploying a Script to the Database from the Command Line

See Also

- Getting Started with Script Author
- Working with Graphical Script Files
- Working with Script Author Toolbars
- Viewing Objects on the Canvas
- Working with Objects on the Canvas
- Working with Branches on the Canvas
- Defining Global Script Attributes
- Defining Panels
- Defining Groups
- Defining Blocks
- Defining Branches
- Defining Database Connections
- Controlling Script Flow
- Using the Panel Layout Editor
- Defining Panel Question UI Controls
- Working with Questions
- Defining Actions
- Defining Commands
- Reusing Commands

4.20.1 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.

See Also

- Deploying a Script to the Database from Script Author
- Deploying a Script to the Database from the Command Line

4.20.2 Deploying a Script 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.)

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.

References

- Using the Scripting Engine Web Interface
- Defining the Script Name

See Also

- Checking Script Syntax
- Deploying a Script to the Database from the Command Line

4.20.3 Deploying a Script 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.

See Also

- Checking Script Syntax
- Deploying a Script to the Database from Script Author

5

Using the Scripting Engine

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).

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. No other components display, other than the Web browser's own interface controls.

This section includes the following topics:

- Scripting Engine Concepts
- Using the Scripting Engine Agent Interface
- Using the Scripting Engine Web Interface

5.1 Scripting Engine Concepts

This section includes the following topics:

- Oracle Scripting and Oracle Applications Sessions
- Oracle Scripting Sessions
- Script Transactions

5.1.1 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*.

See Also

- Oracle Scripting Sessions
- Script Transactions

5.1.2 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, open 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.

There is no recovery from an expired session; close the Web browser window and re-launch the script in a Web browser if your session times out.

See Also

- Oracle Scripting and Oracle Applications Sessions
- Script Transactions

5.1.3 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 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.

Clicking the Suspend button in the agent interface, for the purposes of this discussion, is considered standard (if temporary) completion of a script session. There is no equivalent for the Disconnect or Suspend buttons in the Web interface.

Upon standard completion of a script, when the Continue 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.

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 manually 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 script transaction is terminated upon reaching the session idle timeout specified on the Apache server. In this case, 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 manually 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.

See Also

- Oracle Scripting and Oracle Applications Sessions
- Oracle Scripting Sessions

5.2 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 11*i* is integrated with three business applications:

- The Customer Support module of Oracle TeleService
- Oracle TeleSales
- Oracle Collections

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:

- Launching a Script in Standalone Mode
- Executing a Script in the Agent Interface
- Suspending an Oracle Scripting Transaction
- Resuming a Suspended Oracle Scripting Transaction

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.

See Also

- Scripting Engine Concepts
- Using the Scripting Engine Web Interface

5.2.1 Launching a Script in Standalone Mode

Use the following procedure to launch a script in the Scripting Engine agent interface in standalone mode. 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 Corporation.

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.

See Also

- Executing a Script in the Agent Interface

- Suspending an Oracle Scripting Transaction
- Resuming a Suspended Oracle Scripting Transaction

5.2.2 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

Scripting Agent, Vision Enterprises

Scripting User

Customer Support

Telesales Agent

Collections Agent

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.

See Also

- Launching a Script in Standalone Mode
- Suspending an Oracle Scripting Transaction
- Resuming a Suspended Oracle Scripting Transaction

5.2.3 Suspending an Oracle Scripting Transaction

Use the following procedure to suspend a current Oracle Scripting transaction running in the Scripting Engine agent interface.

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

Scripting Agent, Vision Enterprises

Scripting User

Steps

1. From the Navigator, select Scripting Demo Form and click **Open**.

   The Script Chooser window appears.

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.

4. Navigate through the script, selecting answer controls as appropriate. To continue to the next panel, click **Continue**.

5. At any point during the script transaction, from the bottom of the Oracle Scripting window frame, click **Suspend.**

   The Oracle Scripting window closes. The script is suspended. To initiate a new Oracle Scripting transaction, open the minimized Script Chooser and select another script as appropriate.

See Also

- Launching a Script in Standalone Mode
- Executing a Script in the Agent Interface
- Resuming a Suspended Oracle Scripting Transaction

5.2.4 Resuming a Suspended Oracle Scripting Transaction

You can resume a suspended Scripting Engine agent interface transaction from Oracle Scripting's Suspended Transactions Chooser form.

Resuming a script using this form is intended for script testing only, and is otherwise not supported by Oracle Corporation.

Prerequisites

- You must execute resumed scripts in standalone mode.
- At least one Oracle Scripting transaction executed in the Scripting Engine agent interface must first be suspended.
- If multiple transactions have been suspended, you must be able to identify the transaction you want to resume.

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 Suspended Transactions Chooser and click **Open**.

   The Restart Suspended Transactions window appears.

2. If you want to list the set of suspended transactions, from the View menu, select **Find All**.

   The Restart Suspended Transactions window refreshes, listing all suspended transactions.

3. If you want to specify the transaction, perform the following:

   a. From the **View** menu, select **Query By Example > Enter**.

      The Restart Suspended Transactions window refreshes, with the first row selected in query mode.

   b. Click in the column representing your search criteria, and type the criteria.

      Note: Scripts are listed in the Suspended Transactions Chooser by transaction ID, Script ID, Script Name, Language, script creator (created by), script end time (the date and time the script transaction was suspended), and segment count.

      For example, to search for a script with the script ID of 218, click in the Script ID column and type 218.

   c. When satisfied with your search criteria, from the **View** menu, select **Query By Example > Run**.

   d. Click in the column representing your search criteria, and type the criteria.

      For example, to search for a script with the script ID of 218, click in the Script ID column and type 218.

      The Restart Suspended Transactions window refreshes, listing all suspended transactions that match your search criteria.

4. Click in the row representing the appropriate Oracle Scripting transaction to resume.

5. Click **Start Script.**

   The Script Chooser automatically selects the appropriate script and minimizes.

   A new window appears, containing the resumed script. State is returned for all parameters to the equivalent state at the point when the script was suspended.

   - The active panel upon resumption is the same panel that was active when the script was suspended.
   - Responses for all questions already answered appear in the Progress area.
   - Blackboard values for any blackboard commands and all answers up to this point in the script are restored to static memory.

- Panel footprinting and answer collection information (if enabled) is enabled.

Guidelines

Scripts are listed in the Suspended Transactions Chooser by transaction ID, Script ID, Script Name, Language, script creator (created by), script end time (the date and time the script transaction was suspended), and segment count.

See Also

- Launching a Script in Standalone Mode
- Executing a Script in the Agent Interface
- Suspending an Oracle Scripting Transaction

5.3 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:

5.3.1 Web Interface Concepts

Survey Campaign Administration Requirements

Survey campaigns are administered in the Survey Administration console by an Oracle Applications user with the Survey Administrator responsibility. In order to execute in a Web browser, a survey campaign must contain a cycle and deployment, and the deployment must be activated. Scripts run in this interface display components known as survey resources, which are also administered by survey administrators. For survey administration information, refer to the Survey Campaign Administration Tasks section in *Oracle Scripting Implementation Guide*.

Web Interface Architecture

From a Web interface perspective, the Scripting Engine is a set of Java classes on the middle tier that receives requests from the JSP layer (including the message to launch a script), and processes them. All business logic (as determined by the survey questionnaire script, custom code or database calls, and respondent actions) are evaluated by the Scripting Engine, which passes processing information to the JSP layer to interpret, which in turn sends information to the Web browser to display.

Web Interface Users

There are three typical sets of users of the Web interface:

- Users who access a script in a Web browser from a self-service application.
- Users who are invited to participate in a survey through an e-mail message.

- Users who access a Web script or survey in a Web browser by manually entering a valid survey URL into the Location or Address field in their Web browser.

All require survey administration, which results in a valid survey URL. For more information, see Launching Scripts in the Web Interface.

Enabling Access to Scripts in the Web Interface

To enable Web application users to access the Web script or survey, the survey URL must be embedded into the application as a hyperlink. This process involves customization of the appropriate Web application's user interface.

To send invitation or reminder e-mail messages inviting members of a known sample or population to execute the script, you must have an Oracle Marketing list, and perform master document, query and template administration using Oracle One-to-One Fulfillment. For specific details on these administration steps for, see the appropriate product documentation. For information specific to Oracle Scripting, you can also refer to the Campaign and List Administration Tasks and the Survey Campaign Administration Tasks sections of *Oracle Scripting Implementation Guide*.

Survey Campaign, Cycle, Deployment Hierarchy

The hierarchy for defining a survey campaign in the Survey Administration console is:

- Survey Campaign
- Cycle
- Deployment

Survey Campaign

The survey campaign is at the top level of the hierarchy. It is where parameters affecting the entire survey campaign and all its "child objects" (the cycle and deployment) are administered. These parameters include the specific script to be used as the survey questionnaire, the survey campaign name, the base technology stack to be used to execute the script at runtime, and which survey resources are used. A survey campaign contains one or more **cycles** (described below).

Cycle

Cycle properties include only a cycle name. Each executable deployment belongs to a parent cycle, and each cycle belongs to a parent survey campaign.

The ability to define multiple cycles in a survey campaign aids in reporting for comparative data analysis for surveys to be conducted over a span of time.

You must have at least one cycle. You must create a cycle at the time of survey campaign creation. You can modify the cycle or create additional cycles afterwards, for any open or active survey campaign.

Deployment

A deployment is the lowest member of the survey hierarchy. It is also the most granular. Deployments belong to a particular parent cycle. Deployment properties include a deployment name, a media type (currently, only WEB is supported), a status, a deploy date and time, a response end date and time, and a deployment type (standard or targeted). For standard deployments, that is the sum total of parameters.

Targeted deployments are intended for execution in coordination with Oracle Marketing (for lists of targeted individuals) and Oracle One-to-One Fulfillment (for sending my e-mail invitations or reminders to participate in the Web-based survey).

On the Create Survey Deployment page, when you select Targeted for deployment type and click **Go**, the page refreshes, showing deployment parameters specific to targeted deployments. These include list parameters, invitation and reminder parameters, an optional field for determining the target percentage of responses to receive before closing the deployment, and batch request details for the concurrent request and the fulfillment request associated with the deployment.

Hosting options are also set on this page, and the basis for the targeted survey Web Uniform Resource Locator (URL) (including the logged-in Apache host and port) is listed. If you wish to execute this script on an Apache Web server or port different from the instance in which you are logged in, you must define this secondary Web server or port here.

Essentially, deployments are the construct within the survey campaign that contain key business rules (the "who," the "when," and the "how long") for that portion of a survey campaign. A deployment must be explicitly made active before respondents can execute the survey (by participating in a survey campaign or executing a Web script).

Survey Media Channels

In current releases of Oracle Scripting, there is a single supported media channel for taking surveys: the World-Wide Web (Web). In the future, other media channels are intended to be supported. For example, a media channel such as "Call Center" would support the business case in which call center agents would input survey

responses provided from respondents over the telephone. In current releases this parameter is hard-coded. When other media channels are available, you will select the appropriate channel at the *deployment* level when administering a survey campaign.

See Also

- Launching Scripts in the Web Interface

5.3.2 Launching Scripts in the Web Interface

This section includes the following topics:

- Launching a Script from an Oracle Self-Service Web Application
- Launching a Script from an Invitation or Reminder
- Launching a Script from a Known URL as a Guest User

See Also

- Web Interface Concepts

5.3.2.1 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.

See Also

- Launching a Script from an Invitation or Reminder
- Launching a Script from a Known URL as a Guest User

5.3.2.2 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.

See Also

- Launching a Script from an Oracle Self-Service Web Application
- Launching a Script from a Known URL as a Guest User

5.3.2.3 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.

See Also

- Launching a Script from an Invitation or Reminder
- Launching a Script from a Known URL as a Guest User

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

Obsolete term. See question.

answer UI type

Obsolete term. See question user interface type.

answer user interface type

Obsolete term. See question user interface type.

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 Corporation 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 11i 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.

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.

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.

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 ERP applications and Forms-based CRM 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.

question UI type

See question user interface type.

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.

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 series of Java components wrapped in an Oracle Form (including 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 (only the panel presentation area is displayed). 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 and graphical scripts. 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 graphical script and wizard script.

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.

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.

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.