Oracle® Scripting

User Guide

Release 11*i*

September 2002

Part No. B10207-01

Oracle Scripting User Guide, Release 11*i*

Part No. B10207-01

Contents

3 Using Oracle Scripting

Send Us Your Comments

Oracle Scripting User Guide, Release 11*i*

Part No. B10207-01

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

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

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

- Postal service:
  Oracle Corporation
  Oracle Scripting Documentation
  1900 Oracle Way
  Reston, Virginia 20190
  U.S.A.

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

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

Preface

Audience for This Guide

Welcome to Release 11*i* of the Oracle® Scripting Concepts and Procedures Manual.

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 three components of Oracle Scripting: Understanding the Script Author, Understanding the Scripting Engine, and Understanding the Survey Component.

- **Planning Oracle Scripting Projects** describes planning aspects required for Oracle Scripting, including planning Scripting Engine projects, and planning Oracle Scripting survey campaigns
- **Using Oracle Scripting** describes the use of the various components of Oracle Scripting from an agent, end-user, developer, administrator, and customer perspective. This section includes Using the Script Author, Troubleshooting the Script Author, Using the Scripting Engine, Taking a Survey, and Using the Survey Admininstration Console.

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 Corporation 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 Corporation does not own or control. Oracle Corporation 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*. Oracle Scripting online help is only available with installation of Oracle Scripting release 11.5.6 or later.

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 Implementation Guide 11*i*

This guide describes how to implement Oracle Scripting components and test the implementation appropriately.

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 API's 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 Application Foundation Implementation Guide

Many CRM products use components from CRM Application Foundation. Use this guide to correctly implement CRM Application Foundation.

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 Appl i cati ons. 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 topic group includes the following topics:

Oracle Scripting Overview

Understanding the Script Author

Understanding the Scripting Engine

Understanding the Survey Component

1.1 Oracle Scripting Overview

This topic group includes the following topics:

What Is Oracle Scripting?

Why Use Oracle Scripting?

Who Are the End Users of Oracle Scripting?

Why Is Oracle Scripting in the Interaction Center Suite?

See Also

Understanding the Script Author

Understanding the Scripting Engine

Understanding the Survey Component

1.1.1 What Is Oracle Scripting?

Oracle Scripting is a set of tools to create and display text, prompts, and images to end users. The information displayed is based (1) on a predetermined set of requirements that have been programmed into a script, and (2) on responses provided by the end user. End users are guided through a specific flow, which can either be rigidly prescribed to enforce consistency, or which can be dynamically determined based on answers entered into the graphic user interface (GUI).

Oracle Scripting is composed of three components. Each of these are described briefly below. For more information, see the Understanding section for each component.

Script Author

The Script Author component (hereafter referred to as the Script Author) is a powerful authoring environment of graphical layout tools to create, modify, and deploy scripts. The Script Author is a two-tiered script development studio built in Java. The script developer inserts objects into the script in a work area known as the canvas, using an intuitive interface. The developer then associates properties to each object. These properties range from the simple (object names, labels, or text to display at runtime) to the complex (actions containing Java, PL/SQL, or other commands). The objects in the Script Author are linked by various branch types to provide a graph that looks similar to a flowchart. In this manner, the objects embedded in the script control complicated processing, such as rules-based branching and data integration. Scripts built in the Script Author can be run in either interface of the Scripting Engine.

Scripting Engine

The Scripting Engine is a set of base Java classes that process a Script Author script in runtime. The script provides a customized set of pre-programmed actions to take based in part on responses provided by the end user. At runtime, the Scripting engine interprets instructions in the script, the end user’s responses to questions and answers, and custom code referenced in the script.

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 the Script Author. Based on the type of command, the code may be stored in the metadata of the script itself (e.g., constant commands or PL/SQL commands), in the Scripting session (blackboard commands), in custom code deployed to the applications server (Java commands), or in the database (commands referencing stored PL/SQL packages).

There are two runtime interfaces, both of which receive information from the base Java methods that together comprise the Scripting Engine.

The first is the **interaction center agent user interface** (agent interface). This Java-based user interface is started from an Oracle Applications session when an agent starts a script – typically from within one of three Oracle business applications: Oracle TeleSales, Oracle Collections, or the Customer Support module of Oracle TeleService. Agents can also launch a script in "stand-alone" mode from within an Oracle Applications session, although this is recommended primarily for script testing. In current releases, scripts viewed in this model present as a Java bean wrapped in an Oracle form. HTML interpretation and rendering is provided by Java classes in the SWING library of JavaSoft's Java Development Kit (JDK) 1.1.8.

The second interface is the **Survey runtime user interface** (Web interface), which is essentially HTML running Java Server Pages (JSP) in an Oracle Applications 11i-compliant Web browser. This runtime interface is intended for individuals to view a script to participate in a survey, using their Web browser to access a particular URL. This survey URL specifies a set of parameters that identify an enterprise's Oracle Applications-specific Apache Web server, invoke a JSP page for survey operations, specify a deployment identification code (or dID), and (for list-based survey campaigns), specify a unique respondent identification code (rID). This model also requires the user to start an authenticated Oracle Applicaitons session. When taking a survey by visiting a URL sent in an invitation or e-mail reminder, or when coming across the survey URL on an enterprise’s Web site, the Oracle Applications session is initiated by using a guest user login with survey-specific privileges. For individuals taking a survey from an integrated application such as Oracle iSupport, the Oracle Applications login information for the current valid Oracle Applications session is used, and the individual selects a link from iSupport to take the survey. In either model, the Scripting Engine interprets the respondent's responses, the script, and custom code, passing metadata and instructions to the Web browser to be interpreted on the respondent's client computer. In addition to a script, survey execution of scripts requires a survey campaign to be established and deployed, and requires the presence of additional JSP resources (to provide a header for each survey page, an error page to display if required, and a final page after completion of the survey).

Of the objects created in the Script Author, only the contents of a panel display in runtime in the Scripting Engine (the other objects control processing). Each panel contains, at minimum, one answer definition, which displays the appropriate answer control at runtime. Panels may include any number of answer definitions. At the script developer’s discretion, the WYSIWYG (What You See Is What You Get) interface allows liberally customized display of formatted text and images to appear in a panel, interspersed (if desired) between answer controls. 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 any other answer controls in that panel (radio buttons, checkboxes, dropdown or multiselect lists, or text in a text field, text area, or password field), and progresses the script to the next panel.

Survey Component

Using the Survey component of Oracle Scripting, a survey questionnaire (in the form of a script developed using the Script Author) can be made available for people to take using an Oracle Applications 11i-compliant Web browser. Regardless of the entry point, the individual taking the survey (the respondent) will view the script in a Web browser running JSP pages on the Apache Web server of the polling enterprise. (For more information on entry points, see Survey Respondent Entry Points in the Using Oracle Scripting section of this document.)

In order for this script to be made available as a survey questionnaire, a survey campaign must be defined by an administrator using the Survey Admininstration console. The purpose for executing scripts as a survey campaign is to obtain a specific set of data from the target population (or a general population). This data is typically analyzed by the polling enterprise, and actions can be taken as a result of the survey campaign. These actions are directly dependent on the purpose for conducting the campaign and the population surveyed. For example, a large company polling its employees might be interested in knowing how satisfied the employees are with the current benefits or compensation packages currently available. Typical uses of a survey campaign are to measure political opinion; to measure satisfaction; to determine whether a set of products or services meets the needs of a target population; or to perform market research to determine the viability of new products or services.

See Also

Survey Respondent Entry Points

Why Use Oracle Scripting?

Who Are the End Users of Oracle Scripting?

Why Is Oracle Scripting in the Interaction Center Suite?

1.1.2 Why Use Oracle Scripting?

The Script Author is used to develop scripts for execution in either the Scripting Engine or as a survey using the Survey component of Oracle Scripting.

The Scripting Engine agent interface can be used in inbound, outbound, or blended interaction centers in order to guide agents in 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. The Scripting Engine Web interface can be used by any 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.

The Survey component of Oracle Scripting is used to enforce the business rules regarding survey campaign requirements: selecting the appropriate (deployed) survey questionnaire script and global survey resources to be used, designating the media type, specifying start and end dates and times for executing each deployment, and for list-based campaigns specifying the list, e-mail invitation or reminder master documents, and enforcing uniqueness of responses from designated respondents. The Survey component also allows administration in terms of monitoring specific responses or analyzing compiled data from surveys taken. Four out-of-the-box reports are available to assist in analysis of data obtained by executing Oracle Scripting. All are applicable to survey campaigns, and the Footprinting report is useful for interaction center reporting for the agent interface. More reports are expected to be available in future releases. The information obtained from a survey is stored in the Oracle Applications database, and can be accessed using customized SQL scripts or by using any tool that can access an Oracle database. Additional survey-specific data is stored in summary tables, and can be loaded and normalized using concurrent programs.

See Also

What Is Oracle Scripting?

Who Are the End Users of Oracle Scripting?

Why Is Oracle Scripting in the Interaction Center Suite?

1.1.3 Who Are the End Users of Oracle Scripting?

This topic group includes the following topics:

- Script Author Users
- Scripting Engine Users
- Survey Runtime Users

- Survey Administrative Users

1.1.3.1 Script Author Users

The Script Author is intended for the functional user with some technical knowledge. The Script Author development environment provides for reuse of defined commands and existing Scripting components. The script developer will define these commands. Hooks in the GUI reference technical components (for example, Oracle Forms, custom Java methods, and PL/SQL packages stored in the database) which are primarily developed external to the Script Author. 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 on the server and 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 and the Survey component at runtime.

Users of the Script Author may be campaign administrators, Java developers or database programmers, business process engineers, or 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.

In current releases, no Oracle Applications responsibilities are required to use the Script Author, as this is an independent Java environment not requiring login to an Oracle Applications session. In future releases, the Script Author will be launched from an existing authenticated Oracle Applications session of a user with Scripting administrator responsibilities, using Oracle JInitiator.

See Also

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

1.1.3.2 Scripting Engine Agent Interface Users

End users of the Scripting Engine agent interface are trained interaction center agents. These are non-technical users who have received simple but thorough training in the Java-based Scripting Engine agent 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. These agents typically have access to (and training in) at least one Oracle business application integrated with Oracle Scripting. While the Scripting Engine agent interface can launch scripts in "stand-alone mode" *without* an integrated application (for example, to test scripts), the agent interface is intended to be used in combination with integrated Oracle business applications to take full advantage of the contact handling and service, sales, or collections features of those applications, respectively.

The table below lists the Oracle Applications and associated responsibilities that can currently launch the Scripting Engine.

Responsibilities That Can Launch the Scripting Engine Agent Interface

Table 1–1 Oracle Applications Responsibilities That Can Launch the Scripting Engine

Responsibility Name	Application	Purpose
Customer Support	Oracle TeleService	Enable a regular Customer Support (TeleService) agent to launch a script
TeleSales Agent	Oracle TeleSales	Enable a regular TeleSales agent to launch a script
Collections Agent	Oracle Collections	Enable a regular Collections agent to launch a script
Scripting User	Scripting Engine	Launch Scripting Engine in stand-alone mode
Scripting Agent, Vision Enterprises	Scripting Engine	Launch Scripting Engine in stand-alone mode (for environments without Scripting User responsibility)

References

- For information about establishing responsibilities, refer to the *Oracle Applications System Administrator's Guide Release 11i,* in the section Managing Oracle Applications Security > Defining a Responsibility.
- For a list of Oracle Applications responsibilities that may be required for use with Oracle Scripting, refer to the *Oracle Scripting Implementation Guide, Release 11i,* in the section Creating and Administering Users.

- For information about application-specific responsibilities, please refer to the Implementation Guide for each specific business application.

See Also

- Script Author Users
- Survey Runtime Users
- Survey Administrative Users

1.1.3.3 Survey Runtime Users

End users of the Survey component of Oracle Scripting from a runtime perspective are customers or prospects viewing a script (typically a survey questionnaire) in any Oracle Applications-compliant Web browser. These may have been invited to participate in a survey through an e-mail message or through navigation to a survey-enabled site, or they may have accessed the survey through a self-service Web application scenario such as Oracle iSupport.

To execute a script as a survey in a Web browser, survey respondents accessing a survey URL on an enterprise Web site or responding to an e-mail invitation or reminder use the guest login established for that environment. Survey respondents accessing surveys from Oracle iSupport use the login information for their currently authenticated Oracle Applications session.

See Also

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

1.1.3.4 Survey Administrative Users

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

To access the Survey Administration console HTML user interface, these users must have the Survey Administrator responsibility.

See Also

What Is Oracle Scripting?

Why Use Oracle Scripting?

Why Is Oracle Scripting in the Interaction Center Suite?

1.1.4 Why Is Oracle Scripting in the Interaction Center Suite?

The Scripting Engine provides enterprises 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 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. The Survey component is a powerful information gathering tool that allows rapid deployment of tailored survey campaigns using questionnaires to gather data from various 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.

See Also

What Is Oracle Scripting?

Why Use Oracle Scripting?

Who Are the End Users of Oracle Scripting?

1.2 Understanding the Script Author

The Script Author component of Oracle Scripting is a powerful authoring environment of graphical layout tools to create, modify, and deploy scripts. The Script Author is a two-tiered script development tool supported on Windows clients (Windows 95, Windows 98, Windows NT, and Windows 2000 operating systems) for creating scripts. These scripts are deployed to the Oracle Applications database and can be executed in the Java-based Scripting Engine (in an interaction center) or as a survey (running as a JSP page in a Web browser).

This topic group includes the following topics:

How to Obtain the Script Author

What Is a Script?

What Are the Minimum Requirements for Any Graph?

Understanding Answers

See Also

Oracle Scripting Overview

Understanding the Scripting Engine

Understanding the Survey Component

1.2.1 How to Obtain the Script Author

The Script Author software is laid down on the APPL_TOP of the Oracle Applications server during a Rapid Install or upgrade, specifically in the product top for Oracle Scripting (product code IES).

Prerequisites

- Oracle Applications 11.5.6 or later must have been installed using a Rapid Install at your enterprise or site.
- The Script Author must be installed on a Windows (95, 98, NT, 2000) machine.

Prerequisites

- You must be using an Oracle Applications 11*i*-compliant Web browser.
- You must know the appropriate URL to access your Oracle Applications environment.
- You must have enough space allocated on the target computer to download and extract the Script Author files. The compressed file is approximately 80 megabytes in size. This size may vary on your local machine.

Login

None required

Responsibility

None required

Steps

1. From your Web browser, access the appropriate URL for your environment, using the following syntax, and press Enter.

   ```
   http://<server>.<domain>:<port>/OA_HTML/download/ies/iesauthr.zip.
   ```

   Fill in the server, domain, and port specific to your environment. For example:

   ```
   http://server1.yourcompany.com:7777/OA_HTML/download/ies/iesauthr.zip
   ```

2. Your Web browser will prompt you to perform an action with the specified file. Select the option that allows you to save the file to your local filing system.

   a. If you are downloading the files using the target computer, navigate to the desired directory and save the file.

      After installation, you do not need to retain the installation package, since it will be accessible from the same URL at any time. Therefore, you may save the installation package in any location, including your default TEMP directory.

   b. If you are downloading the files for installation on a different computer, save the file to any location, and transfer the ZIP file to the target computer.

3. Decompress the installation files.
4. Install the Script Author using the Oracle Universal Installer.
5. When running the Script Author for the first time, you may be prompted to either load strings from the database or install default. For more information, see Troubleshooting the Script Author > Storing Strings in the Script Author.

See Also

Storing Strings in the Script Author

What Is a Script?

What Are the Minimum Requirements for Any Graph?

1.2.2 What Is a Script?

This topic group includes the following topics:

- Configurable Objects
- Non-Configurable Objects

- Branches

From a Script Author perspective, a script is basically a graph made up of configurable and non-configurable 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 script contains, at minimum, one graph. This is called the root graph (the graph which is visible when you open any 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.

From an Oracle Applications perspective, a script is a file (created in the Script Author, saved as FILENAME.SCRIPT or FILENAME.SCR, and deployed to the applications database) that can be executed at runtime in the Scripting Engine component in one of two interfaces: the agent interface, a Java-based UI for interaction center agents, and the Web interface, in which the script runs in a Web browser (generally for survey respondents). When the script is run, it displays text, prompts, and images to 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.

See Also

How to Obtain the Script Author

What Are the Minimum Requirements for Any Graph?

Understanding Answers

1.2.2.1 Configurable Objects

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

- Panels
- Groups
- Blocks

See Also

- Non-Configurable Objects
- Branches

1.2.2.1.1 Panels A **panel** object contains the information that is displayed in Oracle Scripting Engine at runtime. Answer controls defined in the panel (using the Script Author) are used to accept information from the user at runtime and advance the flow of the script. This is the only object to display at runtime; the other objects are processing units only.

1.2.2.1.2 Groups A **group** object 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). It must certain requirements (see What Are the 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, so that in runtime a Shortcut button can appear in the GUI that will "jump" the user to that group
- Restrict the access of certain or all users to a section of the script at runtime

In addition, when a saved script is imported into a new script, it is imported as a group.

1.2.2.1.3 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 command that references the API.

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

A block represents a child graph (a subgraph). It must meet certain requirements (see What Are the Minimum Requirements for Any Graph?).

1.2.3 Non-Configurable Objects

There are two *non-configurable* objects in Oracle Scripting Author. These objects are:

- Start node
- Termination node

See Also

- Configurable Objects

- Branches

1.2.3.1 Start Nodes

Every time a graph is created in the Script Author, it contains a Start node. Start nodes are non-configurable. They 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 branch to the first explicitly created object in the script.

1.2.3.2 Termination Nodes

Termination nodes are required on every graph. They represent the end of the flow of that level of the script in runtime. They are non-configurable, containing no viewable properties. Every graph must contain at least one Termination node, attached with branches to objects on the canvas. A graph may contain two or more Termination nodes, each depicting the end of a flow of a particular path.

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.

1.2.4 Branches

A branch is used to connect objects. The type of branch used 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 the Script Author.

- **Distinct:** The destination is based on the answer selected in a panel at runtime.

  This branch type requires lookup values to be defined in the panel, and requires values to be assigned to each distinct branch from the branch properties in the Script Author.

- **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 the Script Author and corresponding Java code to provide the expression to be evaluated 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 command to be associated with the branch in the Script Author and corresponding Java code (to provide the expression and destination objects) to be evaluated at runtime. The script can then flow to any sibling object (panel, block or group on the same canvas) or a destination group (using the Shortcut property) on any level of the script.

Branches and Branch Order

Below the Start node, each object placed on the Script Author canvas 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 dialog.

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

- Configurable Objects
- Non-Configurable Objects

1.2.5 What Are the Minimum Requirements for Any Graph?

A Script Author 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 non-configurable objects will also be connected with branching as appropriate.

The Shortest Syntactically Correct Script

To be syntactically correct, a script (or any graph on any level of a 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 answer 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 answer user interface controls in a script. At minimum, each panel has one answer control (a button, often labeled "**Continue**"). If multiple nodes or answer definitions are programmed into a panel, then there will be one answer UI control per definition, in addition to a Continue button.

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

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

How to Obtain the Script Author

What Is a Script?

Understanding Answers

1.2.6 Understanding Answers

Every panel requires at least one "node" or answer to be defined in order for the script to be syntactically correct. This renders in runtime as an answer control in the panel that requires end user input before displaying the next panel. At minimum, for each panel, the end user must click the "Continue" button to progress to the next panel. Some answer definitions, based on the selected UI type, may require additional end user input. For example, a radio button requires the end user to select one of the displayed radio buttons prior to accepting the "Continue" request and progressing to the next panel.

How Many Answer Definitions Are Required Per Panel?

A panel must have at least one answer definition, and may have any number of answers defined. There are only two limits: practicality and distinct branching. At runtime, the user should be able to easily navigate each panel. The more answers, the more extensive the panel layout; users generally prefer to view the contents of a panel in a single screen. Regarding distinct branching, only one answer per panel can directly affect distinct branching. If a panel uses distinct branching for an outgoing branch type, the answer definition that uses distinct branching *must* have the "Default for Distinct Branching" option selected.

Answer Definition Properties

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

Property	Data Type	Required?	Function or Description	Restrictions
Default for Distinct Branching	Check box	No	Indicates whether this answer definition 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 answer definition per panel may have this option selected.
Name	Text	Yes	Identifies the specific answer definition to the Oracle Scripting application. When a value is provided at runtime, this value is the key to access the answer value provided by the application end user.	Supports any text in the current character set. For each panel, the panel name should be unique.
UI Type	Drop-down List	Yes (defaults to Text Field UI type if no selection made)	Identifies the answer control which renders at runtime for end user input. Choices include Text Field, Text Area, Radio Button, Check Box, Button, Drop Down, Password, Checkbox Group, and Multi-Select List Box.	One UI type per answer definition.
Label for reporting	Text	Only for single Check Box UI	For all UI types except check box, entering text in the label for reporting results in that text string appearing as a label to the left of the answer in runtime. This label also appears in reports from the HTML administration console	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 an answer, 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 an answer, is no longer supported due to conflicts with the WYSIWYG architecture.	Not functional with Oracle Scripting 11.5.6 and later.

Required Answer Definition Properties

- The properties required depend in part upon the selected answer UI type. The only required property for all answers is Name. If you create an answer definition 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 answer definitions to be defined as the trigger for distinct branching. Distinct branching cannot be triggered on panels containing multi-select answers, since more than one of the lookup values may be selected. Thus, this property is unavailable to the multiple-selection answer UI types.
- Panels containing radio button, button, drop down, checkbox group, and multi-select list box answer UI types all require lookup values to be defined by the script developer. Lookup values include a table lookup, cursor lookup, or command lookup. These are accessed in the data dictionary for a specific answer definition by accessing **Answer Entry Dialog > Data Dictionary > Lookups**.

WYSIWYG Architecture Restrictions

1. Once the answer definition properties listed above are saved, changes to answer properties behave in the following manner in Script Author:

- Changes to the Default for Distinct Branching option *will* be recognized, unless the UI Type is one of the multi-select options.
- Changes to the Name property *will* be understood by the Scripting Engine regardless of execution interface.
- The UI Type list is not modifyable from the Answer Entry dialog box.
- Although the Label for reporting field is modifyable in the Answer Entry dialog box, any changes made here after the first save committed *will not be reflected in the panel layout*.

If you want to change one of the unmodifyable Answer Entry dialog box properties such as UI type or Label for reporting, you must take one of two 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 the Script Author, re-import the panel text, replacing the previous answer definition of the same name.

   b. Create a new answer definition, applying the appropriate properties. If you wish to provide the newly defined answer with the same answer name, you must delete the original answer definition prior to checking script syntax, or you will encounter the error message "error: duplicate answer name <answer definition name>".

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 box. Replacement

of answers 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 using undocumented and currently unsupported JarLoader tools.

Answer User Interface Types

The Script Author provides nine Answer UI types, consisting of a familiar set of answer controls that render in HTML forms: buttons, checkboxes, radio buttons, text fields, text areas, and password fields. However, some of the characteristics of each may act differently than in a standard HTML page. The table below briefly describes each type.

UI Type	End User Action	Supports Null Value?	Supports Multi-Select Values?	Requires Lookup Value?
Text Field	Keyboard entry	Yes	No	No
Text Area	Keyboard entry	Yes	No	No
Radio Button	Click	No	No	Yes
Check Box	Click	Yes	No	Yes
Button	Click	No	No	Yes
Drop Down	Click, drag, click	No	No	Yes
Password	Keyboard entry	Yes	No	No
Checkbox Group	Click	Yes	Yes	Yes
Multi-Select List Box	Click, Ctrl-Click (Option-Click for Macintosh OS)	Yes	Yes	Yes

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

- Text field, text area, and password field answer 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 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 lookup values to be selected, assuming no default command is defined for the panela answer.
- Both multi-select answer UI types (checkbox groups and multi-select list boxes) can accept null (unselected) as a valid answer.
- Both checkbox types (single and checkbox groups) can accept null (unselected).

Guidance on Using Answer UI Types

Guidance on when to use specific answer 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 answer 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.

UI Type	Description and Recommendations for Use
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: <answername> 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 answer control differs from all others in this respect: all other click-based answer UI types require one or more choices to be defined as lookup values. This UI type will not display lookup values. Thus, label is an absolute requirement for this answer UI type. This answer 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."
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 answer UI type do not contain the automatically generated "Continue" button that appears with all other UI types. A button answer definition 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 the 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 cannot be used with this answer UI type as it will not display in runtime. This answer UI type can only be used in a panel with a single answer definition. A panel with the button UI answer type cannot contain any other answer 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 "<answername>": 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.

UI Type	Description and Recommendations for Use
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 answer definition 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 answer definition’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.
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 answer 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 each lookup value defined, a separate check box appears 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. Each selected checkbox in the group at runtime passes a value of "true" and an unselected checkbox passes a value of "false." These values are stored as a vector of answers. Thirdly, the Label for reporting will function 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 answer UI controls. Finally, this answer control does not allow the "Default for distinct branching" property. Distinct branching cannot be triggered on panels containing multi-select answers, since more than one of the lookup values may be selected. Correspondingly, the Default for distinct branching control in the Answer Entry dialog box will dim once this answer 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 answer control does not allow the "Default for distinct branching" property. Distinct branching cannot be triggered on panels containing multi-select answers, since more than one of the lookup values may be selected. Correspondingly, the Default for distinct branching control in the Answer Entry dialog box will dim once this answer UI type is selected, indicating its unavailability. For this answer UI type, the lookup values available for end user response are all visible without user action.

See Also

How to Obtain the Script Author

What Is a Script?

What Are the Minimum Requirements for Any Graph?

1.3 Understanding the Scripting Engine

The Scripting Engine is a set of base Java classes that process a Script Author script and related code in runtime. The Scripting Engine displays text, graphics, and prompts to the end user, interpreting the requirements programmed into the script dynamically at runtime.

Based on those requirements (panel contents, which determine what text, graphics, and prompts display; commands, which can be associated with most objects in the script; and branching logic), the end user's responses control the flow of the script.

For example, one script may rigidly enforce a specific flow, ensuring the same information is relayed to the end user each time a script is run, regardless of the answers provided. Alternatively, for a different script that makes substantial use of branching logic, different responses to the same questions may branch each end user to a different set of content, or to content in a different order. Some users of such a script may navigate through a long path of questions and answers, whereas other users may see the text, graphics, and prompts only for a few panels before the script session is complete.

Each separate run through a script's flow represents a distinct Oracle Scripting interaction, starting with the launch of the script and ending at completion of the script. Script completion in runtime corresponds to a path that results in reaching a Termination node on the root graph in the Script Author.

The Scripting Engine has two runtime interfaces. The same script can run in either interface. Both interfaces:

- Display only the text, graphics, and prompts or answer controls defined in panels.
- Enforce any branching logic.
- Determine the flow of the script dynamically at runtime, based on a combination of interpreting the end user's responses against instructions provided in branching logic, and the execution of any custom code referenced in the script.

This topic group includes the following topics:

See Also

Oracle Scripting Overview

Understanding the Script Author

Understanding the Survey Component

1.3.1 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 "stand-alone" 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 topic group includes the following topics:

The Progress Area

The Script Information Area

The Shortcut Button Bar

The Disconnect Button

The Toolbar

See Also

Oracle Scripting Overview

Understanding the Script Author

Understanding the Survey Component

1.3.1.1 The Panel Display Area

The largest visible region is the panel display area, where agents will see any panel text and answer controls for the one or more answer definitions 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 answer 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 answer control or controls displaying in the current panel. Answer controls supported in the Scripting Engine include familiar answer 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 answer controls on the panel, the agent must click the **Continue** button. The single exception is the case of a button control. A button answer control, *may* contain a value other than "Continue." Nonetheless, selecting that button will progress you to the next panel.

Note: If the script developer uses the "button" answer control UI type in a panel, that panel must contain only a single answer definition (or "node").

See Also

The Progress Area

The Script Information Area

The Shortcut Button Bar

The Disconnect Button

The Toolbar

1.3.1.2 The Progress Area

As the agent cycles through panels, the sequence panels were accessed, the name given to each panel, the name given to each answer definition in the panel, and the anwer (or answers, for multi-select answer definitions) provided at runtime are listed in the Progress frame on the right. As panels are visited in the script, information for each panel appears in the progress area. To the right of the progress area, a scroll bar appears 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. Under multi-panel display mode, the active panel will have focus, and all text and answer 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, the agent can scroll up through the panel information in the progress area (if necessary) using the scroll bar, and click on any previously displayed panel. The selected panel will again populate in the panel display area. In multi-panel display mode, the selected panel will again gain focus. In single-panel display mode, the selected panel will again populate the entire panel display area.

By design, if an agent revisits a previous panel and changes the answer, if branching in the script should 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 flow taken 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 Bar

The Disconnect Button

The Toolbar

1.3.1.3 The Script Information Area

Optionally, information may be associated with a script in the area known as the **Script Information Area**. This is formerly known as the Static Information Panel. The Script Information Area can accommodate up to nine separate pieces of information (text or a timer), including a label for each. These 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.

See Also

The Panel Display Area

The Progress Area

The Shortcut Button Bar

The Disconnect Button

The Toolbar

1.3.1.4 The Shortcut Button Bar

The **shortcut button bar** appears at runtime 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 provide the agent with the ability to "jump" from the current panel to a specified group elsewhere in the script, or to open a browser window populated with a specified Web page. In order to jump to a group in the script, you must associate a Java command with the shortcut. The target group must have a Shortcut name, and this name must be referenced by the Java command.

The function, display content, and tool tips for buttons are global script properties. In the agent interface, they will appear in every session of that script. Button information is added from the Shortcut Panel in the script properties dialog.

Buttons can be enabled or disabled dynamically at any point in a script using Scripting APIs.

See Also

The Panel Display Area

The Progress Area

The Script Information Area

The Disconnect Button

The Toolbar

1.3.1.5 The Disconnect Button

Optionally, a **Disconnect** button which appears on the bottom right of the Scripting Engine window can be associated with a specific group (a "Wrapup group" to capture information from any individual at the close of the transaction) or it can be enabled to end the script without displaying information. This is programmed using the WrapUpShortcut property of a group in the Script Author as part of the script customization.

See Also

The Panel Display Area

The Progress Area

The Script Information Area

The Shortcut Button Bar

The Toolbar

1.3.1.6 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 will open 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 Bar

The Disconnect Button

1.3.2 Web Interface

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

At this time, the primary purpose of the Web interface is to run a script as an online survey questionnaire to meet the requirements of a survey campaign administered in the Survey component of Oracle Scripting. Individuals that wish to respond to a Web-based survey visit a specific survey URL.

Executing a survey requires an active, authenticated Oracle Applications session. If the survey respondent is already logged into an appliction 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 a guest user login established for this purpose.

At this time, use of the Web interface requires implementation of the Survey component of Oracle Scripting, since a survey campaign, cycle, and deployment must be created and activated before a script can be executed in the Web interface.

The business logic built into the script, including custom Java, PL/SQL, and other commands, guides the survey respondent through the script. Each panel of the script (deployed to the applications server of the polling enterprise) displays as a separate HTML page. This equates to running the agent interface in single panel display mode. Unlike the agent interface, the Web interface does not support multi-panel display mode. Since each panel renders as a separate page, survey respondents cannot display previously visited panels on the same page. However, a survey respondent can use the Back button of the Web browser to visit previous panels, and can change answers if desired.

As the survey respondent responds to the answer 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 respondent, progressing or flowing through the script in a logical manner and accomplishing the exchange of information required by the polling enterprise.

This topic group includes the following topics:

1.3.2.1 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 bar, or disconnect button. Instead of the Oracle Applications toolbar, the browser's default toolbar appears.

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

See Also

Ramifications of User Interface Differences

Survey Resources

1.3.2.2 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 bar 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.

See Also

User Interface Components Differ

Survey Resources

1.3.2.3 Survey Resources

When creating a survey campaign in the Survey component of Oracle Scripting, one prerequisite is to define JSP resources in the Survey Resources subtab under the Survey Campaign tab of the Survey Administration console. At runtime, these must match with physical JSP files of the same name loaded in the $OA_HTML directory on the applications server. Three resources are listed:

- Header Page
- Final Page
- Error Page

See Also

User Interface Components Differ

Ramifications of User Interface Differences

1.3.2.3.1 Header Page For any survey viewed in the Scripting Engine Web interface, there is a header included on every page. This header may include text and graphics. The contents of the header are determined by the physical JSP file referenced as the header page.

1.3.2.3.2 Final Page Upon completion of a survey (after an answer is provided for the last required answer in the last panel), the contents of the physical JSP file referenced as the final page resource is displayed. This typically thanks the survey respondent and contains hyperlinks to the home page of the polling enterprise, or other content at the enterprise's discretion.

1.3.2.3.3 Error Page Should an error in processing the survey occur, the contents of the physical JSP file referenced as the error page resource is displayed.

Hosted surveys, for integration with Oracle iSupport and similar products, use a different error page, known as the hosted survey error page.

1.4 Understanding the Survey Component

The Survey component of Oracle Scripting was previously known as iSurvey. Using this component, an enterprise can easily create and deploy a survey campaign, with which the enterprise polls a population to obtain specific data, typically for subsequent analysis and ultimately, action.

Use of the Survey component of Oracle Scripting is based on a set of predetermined requirements that define a survey campaign. To implement the campaign, you must develop a script to serve as the survey questionnaire and administer a survey campaign.

This topic group includes the following topics:

The Survey Campaign

The Survey Questionnaire

The Survey Admininstration Console

The Survey URL

See Also

Oracle Scripting Overview

Understanding the Script Author

Understanding the Scripting Engine

Understanding the Survey Component

1.4.1 The Survey Campaign

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 Online uses a concept of a campaign that is required by business applications such as Oracle TeleSales, Oracle Collections, Oracle Advanced Inbound and Advanced Outbound.

Similarly, Oracle Scripting has a separate concept supporting its Survey component called a 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 subsequent action. Typical actions might include the offering of a new product or service, or a change in business processes to improve satisfaction.

The 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. A **cycle** is a small collection of requirements related to media type, lists and reporting. A **deployment** contains specific details regarding execution dates for the survey campaign, the target population, and manner of delivery (lists, e-mail invitation and reminder information, etc.).

See Also

The Survey Questionnaire

The Survey Admininstration Console

The Survey URL

1.4.2 The Survey Questionnaire

The survey questionnaire is a Script Author script created to obtain specific information. Typically it consists of a set of questions to gather feedback, obtain market data, tabulate opinion, measure satisfaction, or otherwise collect willingly provided survey data. End users of the Survey component are customers or prospects viewing a survey questionnaire using any Oracle Applications 11*i*-compliant Web browser. As the individual participating in a survey ("survey respondent") moves effortlessly through each HTML page (corresponding to a single script panel in the 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 questionnaire flow, and yields better data and higher response rates.

In the background, the script (and any custom code) runs as a Java Server Pages (JSP) file in an HTML-based execution GUI, typically on the Apache Web server associated with an enterprise applications server and used by Survey administrators.

The script serving as the survey questionnaire must be completed, tested, and deployed to the applications database prior to defining the survey campaign in the Survey Admininstration Console.

Who Creates a Survey Questionnaire?

Based on existing requirements for a specific business purpose, individuals trained in the use of the Script Author component of Oracle Scripting should be used to develop the script that will be viewed by survey respondents. Since surveys are usually a series of simple questions and answers, surveys can in theory be created by relatively non-technical users who have been trained to use the Script Author. However, because Survey is based on the Oracle Scripting product, it also provides very sophisticated functions which can be extended to survey questionnaires in support of survey campaigns that require complex survey solutions. The same additional skill sets that may be required for complex scripts to be executed in the Scripting Engine are applicable to scripts intended for use by the Survey component of Oracle Scripting, including PL/SQL and custom Java, as well as the Scripting-specific use of Constant and Blackboard commands. In practice, therefore, 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

The Survey Campaign

The Survey Admininstration Console

The Survey URL

1.4.3 The Survey Admininstration Console

Survey administrators must define the survey campaign requirements in an HTML administrative console in order to get the survey questionnaire to its target segments. The Survey Admininstration console is a JSP/HTML-based survey campaign administrative GUI accessed from the Oracle Applications HTML login. In order to log into this console you must be a user with the Survey Administrator responsibility. In this GUI, a survey campaign is created and its dependencies (JSP resources, Script Author script designated as the questionnaire, and so on) are defined.

Additionally, results from a survey (ongoing or completed) can be viewed in various ways from the Survey Admininstration console. Answers provided by respondents are stored in the Oracle Applications database schema. A survey administrator can view individual responses by accessing the **Response** tab in the GUI. Survey respondent data is also summarized for optimum performance and made available to administrators through a set of preexisting Survey reports under the **Analysis** tab. Additionally, custom reports can be generated from tables in the Oracle Applications schema as desired.

All survey campaign information must be available to the survey administrator, including the global name of the script serving as the survey questionnaire and any predefined JSP resources, prior to the administrator defining a survey campaign and creating its child objects (the cycle and the deployment).

See Also

The Survey Campaign

The Survey Questionnaire

The Survey URL

1.4.4 The Survey URL

To participate in a survey, you must access a specifically constructed survey URL using an appropriate Web browser. Typically, the survey respondent clicks a hyperlink, which directs the user to the first page of the survey. This URL can also be accessed by clicking a button or image, if the referring HTML page is hyperlinked to associate the image or button with the proper URL.

Components of a Survey URL

The syntax for a valid survey URL is as follows:

```
http://<server>.<domain>:<Apache Web server port>/OA_HTML/
<survey JSP page>?dID=<deployment ID>&rID=<respondent ID>
```

Each complete survey URL includes:

- the server host name and domain
- the Apache Web server listener port
- the Oracle Applications bin (always OA_HTML)
- the appropriate JSP page for a standard or hosted (menu-based) survey
- the deployment identification (dID) number
- the respondent identification (rID) number (for list-based surveys only).

Two Possible JSP Pages

There are two possible JSP pages:

- Standard list- and non-list-based surveys use IESSVYMAIN.JSP as the Java Server Page from which such surveys are launched.

- Hosted surveys (hosting a survey from an Oracle iStore or Oracle iSupport application) use IESSVYMENUBASED.JSP as the Java Server Page from which such surveys are launched.

Apache Port-Specific Parameters

The parameters following the JSP page designation in the survey URL are specific to the particular configured Apache listener port. These include:

- A specific deployment identification number. In the URL string this is referred to as dID. This dID number is generated upon creation of a deployment in the survey administration console. The dID is stored in the SURVEY_ DEPLOYMENT_ID field of the IES_SVY_DEPLYMENTS_ALL table and can be queried from this location. However, the dID is also displayed (as part of the URL string) in the Deployment Details page when a deployment is deployed, or set to active status. The dID is specific to each Apache listener port.
- For list-based survey campaigns only, a respondent identification number. In the URL string this is referred to as rID. This rID number is only generated if Oracle Marketing Online list members are associated with a survey campaign. Generation of the rID occurs when a deployment is deployed, or set to active status. The rID is stored in the RESPONDENT_ID field of the IES_SVY_LIST_ ENTRIES table. Each rID represents a unique individual on the list. The OMO list comprises the entire pool of potential respondents for a list-based survey campaign. Although the same list can be used again and again, the combination of dID and rID uniquely identifies an individual, for each deployment. This combination is unique to each Apache listener port.

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 non-list-based deployments, the dID is the last parameter in a valid URL.
- List-based survey campaigns 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.

Examples

Examples of valid survey URLs include:

Survey Type	URL Example
Non-list-based	http://vision.com:8888/OA_HTML/iessvymain.jsp?dID=1111
List-based	http://vision.com:8888/OA_HTML/ iessvymain.jsp?dID=1111&rID=232
Hosted	http://vision.com:8888/OA_HTML/ iessvymenubased.jsp?dID=1111

See Also

The Survey Campaign

The Survey Questionnaire

The Survey Admininstration Console

2 Planning Oracle Scripting Projects

Oracle Scripting includes three components: the Script Author, the Scripting Engine, and the Survey component. Any Oracle Scripting project requires a script to be developed using the 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 as a survey (executed an Oracle Applications 11*i*-compliant Web browser as a JSP page) in the Scripting Engine Web interface. The same script can also be executed in both modes.

Appropriate planning for Scripting projects depends on many factors. Chief among them is the manner of execution of the script. In other words, will the script be executed in a call center or interaction center (as a call guide for an inbound, outbound, or blended environment)? Or will the script be used as the survey questionnaire in a survey campaign to gather information from a targeted population? 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 topic group includes the following topics:

Planning Scripting Engine Projects

Planning Oracle Scripting Survey Campaigns

2.1 Planning Scripting Engine Projects

Planning Oracle Scripting projects to be executed in the interaction center is typically the task of a consulting group within an enterprise. This involves gathering detailed requirements for building the Script Author 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 topic group includes the following topics:

Facets of Scripting-Specific Discovery Process

Bringing Together the Layers

Tools to Aid in Scoping and Discovery

Oracle Scripting Discovery Data Worksheet

Oracle Scripting Discovery Checklist Tool

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 the 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.

2.1.5 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 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
   - Undetermined
4. Are scripts required that support languages other than English? If yes, identify the language(s):
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? Is the required flow already flowcharted?

7. 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?
8. Is the required flow already flowcharted?
9. Is the required script based on an existing script or call guide? If yes, how many?

   If appropriate, attach the existing scripts to this document and indicate form of existing scripts (hard copy or electronic file).
10. If so, will any business process reengineering be included as part of this effort? If yes, how many?
11. If appropriate, attach the existing scripts to this document and indicate form of existing scripts:
12. What guidance currently exists from which Oracle Scripting scripts will be developed?
    - Narrative
    - Flowcharting
    - System Requirements Document (SRD)
    - Existing scripts or call guides
13. 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:)
14. Is Interactive Voice Response (IVR) unit information required? If so, is IVR in place?
15. Does CTI currently exist? If so, describe:

16. Is integration with other Oracle Applications required? If so, specify below:
    - Oracle TeleSales
    - Oracle Collections
    - Oracle TeleService
    - Oracle iSupport
    - Other (list:)
17. Is a specified script required to launch from an existing business application?
18. If so, does a campaign already exist in Oracle Marketing Online?
19. For survey campaigns, will required survey campaigns be list-based or non-list-based?
20. For list-based campaigns only: Do appropriate lists already exist in Oracle Marketing Online?
21. For list-based campaigns only: Will invitations only be required, or invitations and reminders?
22. Is the survey campaign intended to be anonymous?
23. Will staff at the implementing enterprise create and maintain its own scripts?
24. **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

Bringing Together the Layers

Tools to Aid in Scoping and Discovery

Oracle Scripting Discovery Checklist Tool

2.1.6 Oracle Scripting Discovery Checklist Tool

First, qualify each call guide or script required using the following checklist. Then, again using the checklist template below, break down the requirements for *each* *distinct functionality within each script* (as represented by Groups, 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

Number of Panels:	*Development Level Assessment*
__ Very Small (1 to 25 panels) __ Small (25 to 40 panels) __ Medium (40 to 100 panels) __ Medium to Large (100 to 200 panels) __ Large (200 to 400 panels) __ Very Large (Over 400 panels)	__ Very Simple __ Simple __ Lower Intermediate __ Upper Intermediate __ Complex __ Very Complexs
Scripting Elements Required: __ Mostly Panels __ Mostly Panels and Groups __ Panels, Groups, and Blocks	
Branching Required: (Check all that apply) __ Default (order or branching of panels does not change based on panel answers, e.g., a survey) __ Conditional (boolean logic) __ Distinct (branching based on answers, with all paths known) __ Indeterminate (no 1:1 relationship between answers and path before this interaction) __ All branch types *Branching info Comments:*	

Database Integration: (Check all that apply) __ No database will be used (skip to next) __ Database integration required __ Custom tables required __ Schema(s) available (if so, attach)	***PL/SQL:*** (Check all that apply) __ No PL/SQL will be used (skip to next) Amount of PL/SQL commands required: __ 1 or 2, __ Several, __ Many Are the PL/SQL statements for IES tables or custom tables? __ IES, __ Custom, __ Both

Technology Required

__ HTML	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)
__ Hyperlinks	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)
__ Graphics	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)

Script Author Commands Required

__ PL/SQL	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)
__ Java	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)
__ Blackboard	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)
__ Constant	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)
__ Other	Currently exists?	Y N	Amount:	__ 1 or 2	__ Several	__ Many (List)

Integration Required?

__ Yes	__ No	With:	__ Forms	__ Reports	__ HR	__ Financials	__CRM	__ Other (list:)

Requires jumping (using indeterminate branches and Java) to other groups? __ Yes __ No

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 using the Survey component of Oracle Scripting is to obtain data (via a clearly defined survey questionnaire) from a population over a specific period of time for a particular business purpose. The Survey component of Oracle Scripting enables an enterprise to rapidly solicit and receive such data at low cost, using a Script Author script as the survey questionnaire. 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. The Survey component of Oracle Scripting includes the ability to create on-demand reports from the Survey Admininstration console to aid in data analysis.

This data may be solicited from a targeted population (using predefined 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 the Survey component of Oracle Scripting.

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.

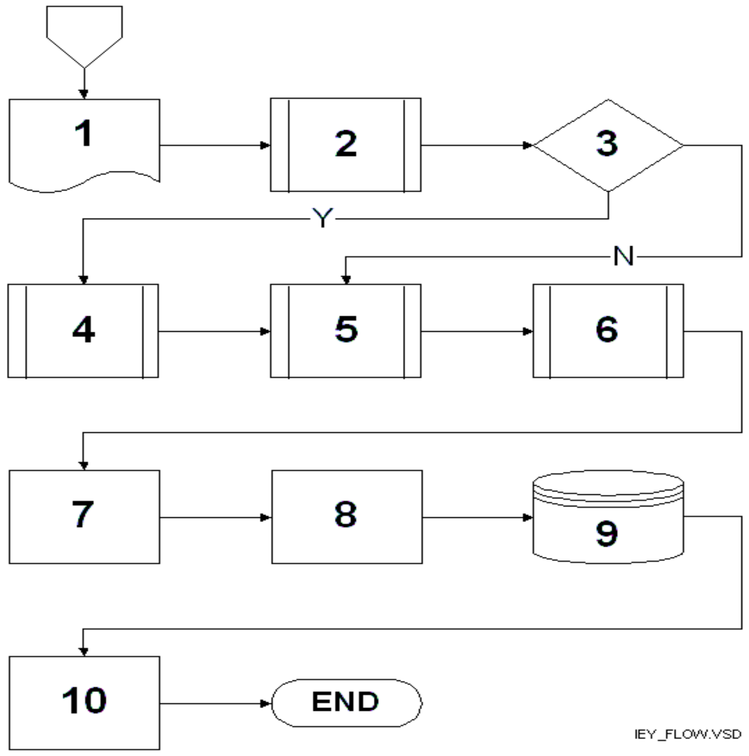

Figure 2–1 Survey Business Process Flow.

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: List-based campaign or non-list-based campaign?

Is survey campaign List-based? If "NO," skip to step 5. If yes, continue to step 4.

4. Generate lists (for list-based survey campaigns).

Generate, create or import lists in the Survey Admininstration 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 Online. Note that successful delivery of invitations (and reminders) to participate in survey campaigns requires fully implemented and configured Oracle One-to-One Fulfillment and Oracle One-to-One Fulfillment plus an outgoing e-mail server.

5. Administer survey resources and survey campaign details.

Create and upload (or modify existing) JSP survey resources to be displayed when using survey campaigns. Then define these survey resources in the Survey Admininstration console. You may establish default resources per user (survey administrator) if desired. The preceding comprise all survey resources administration steps. Subsequently, administer the survey campaign details in the Survey Admininstration console. This includes creating a survey campaign, and defining its subordinate cycle or cycles.

6. Define deployments.

In the Survey Admininstration console, define deployment information for one or more deployments subordinate to each cycle in a survey campaign, based on documented requirements. For list-based campaigns, this will include lists, invitations, and (if included in requirements) reminders.

7. Deploy survey campaign.

In the Survey Admininstration console, set deployments to Active status. This makes the survey campaign available to respondents immediately (or, for list-based 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 Admininstration 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.

Generate on-demand reports from the **Analysis** tab of the Survey Admininstration console, or use SQL-based tools to generate custom reports from the RDBMS. Use concurrent programs to ensure the tabulated data is up-to-date.

Planning

This topic contains the following topic groups:

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for Reporting Survey Campaign Deployment Results

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 Admininstration 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

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for 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. For more information, see the section Survey Respondent Entry Points.
- The requirement for list-based or non-list-based deployments must be indicated. Non-list-based 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 survey questionnaire Script name (the name designated in the Script Author file properties) must be identified.
- The layout, configuration and composition of survey resources (Header, Error Page, and Final Page displayed to survey respondents) must be identified.
- Number of cycles required must be identified, including business rules for each cycle.
- Number of deployments per cycle must be identified.
- Deployment name for each deployment must be identified.
- Business requirement for anonymous or non-anonymous cycles must be determined. See the section Setting Up Invitations and Reminders > Reminders > Reminders and the Anonymous Flag in *Oracle Scripting Implementation Guide, Release 11i.*
- Minimum response percentage for survey cycles must be identified.
- Enterprise Web server and port must be identified.
- Deployment 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 List-Based Survey Campaigns

Additionally, for *list-based survey campaigns,* the following is required:

- Oracle Marketing Online implementation prerequisites must be met. List management portions of OMO must be implemented to make list management features of the Survey Admininstration console available during step 4 and subsequent steps of the business process flow.
- Survey campaign information must be identified down to a deployment level, including list members.
- List member requirements must be identified for each deployment.
- Oracle One-to-One Fulfillment implementation prerequisites must be met. Features that must be available include the ability to create and upload master documents, create queries, and create templates. This requires Fulfillment

implementation, Fulfillment server configuration, and a running Fulfillment Engine.

- Invitation master document layout and content must be determined. These will be created and uploaded when defining deployments (step 6).
- E-mail invitation message subject must be identified.
- Business rules regarding maximum number of responses per respondent must be identified. This is typically set to one (the default value), which ensures a list member can take a survey only one time.
- Requirement for reminders must have been determined.
- Reminder master document layout and content must be determined if reminders are to be employed.
- 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:

Table 2–1 AIM Processes and Two-Letter Process Designations

Acronym	AIM Process
CR	Customer Requirements (Project Management)

Table 2–1 AIM Processes and Two-Letter Process Designations

Acronym	AIM Process
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 Requirements for 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 the 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

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for 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

- The 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 the 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 the Script Author, and copied to the workstation within the enterprise firewall to deploy.

Software

Software for Developing Scripts:

- **Operating system:** Microsoft Windows NT, 95, 98, or 2000.
- Script Author component of Oracle Scripting
- Oracle CRM 11*i*-compliant 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 the Script Author. This is particularly true with survey questionnaires employing primarily sequential logic. However, because the Survey component is based on the Oracle Scripting product, it also provides the ability to include very sophisticated functions in the survey script, if required. Therefore, in order to create adequate survey scripts, you must have received training in using the Script Author tool. Based on the complexity of the script in question, you 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, and so forth.

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

Use of custom Java will also require deployment of custom code to the enterprise server and modification of the JSERV.PROPERTIES file to reference the class path of any custom Java appropriately. This requirement replaces the need to configure the APPSWEB.CFG configuration file in the Caching Architecture for Oracle Scripting. This requires experienced Oracle Applications administrators and will be required any time new JAR files are created in support of a script requiring custom Java classes.

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. They 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 from a workstation within the enterprise firewall. The parameters named below are unique to each environment, as established and maintained by the systems administrator (sysadmin) and the database administrator (DBA). To deploy a script you need:

- Oracle Applications database server host name
- TNS Port number
- Database SID
- apps username and password

For more information, see the section Using the Script Author > Deploying the Script in the *Script Author Online Help*.

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.

Table 2–2

Panel content:	Text and graphics that appear in any HTML page. This includes any information assigned to the "Label for reporting" field when defining a panel answer.
Lookup values:	Predefined answer choices.
Constant commands:	Used to provide *or change* an answer default. In the HTML GUI, the first selection in the range of lookup values becomes the default choice, and checkboxes default to checked or "true." (This is different in the Java GUI.) Use a constant command to provide a different default value for a dropdown, radio button, text field or text area, or to deselect a checkbox as the default.
Validation ranges or requirements:	Validation can be enforced in an answer control. This requires associating a Java method with an answer definition 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.
Branching logic:	If a survey questionnaire is appropriately planned, branching logic can be clearly indicated in a flowchart.
PL/SQL packages:	PL/SQL commands that are loaded in the applications database can be referenced from a script. All such commands must be identified prior to development of the script.
Database table 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.

2.2.3 Requirements for Determining If Survey Campaign Is List-Based

The third step in the Survey business process flow is a decision block, indicating the requirement to determine if the survey campaign is list-based or non-list-based. This determination affects the remaining process flow in terms of order of steps and complexity of campaign setup and administration.

As mentioned in the section Survey Respondent Entry Points, 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 Online capabilities. List-based 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 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.

2.2.3.1 List-Based Campaign = NO

If the ability to execute list-based survey campaigns in an enterprise 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 Fulfillment and OMO are also precluded. The abilities you sacrifice by choosing a *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.

2.2.3.2 List-Based Campaign = YES

When planning to implement or use the Survey component of Oracle Scripting with list-based survey campaigns, the business process flow requires list generation as the next step. For more information refer to Oracle Marketing Online product documentation.

See Also

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for Reporting Survey Campaign Deployment Results

2.2.4 Requirements for Generating Lists

Step 4 of the Survey process flow is the generation of lists using Oracle Marketing Online functionality. 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.

See Also

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for Reporting Survey Campaign Deployment Results

2.2.4.1 Additional Implementation Requirements for List-Based Survey Campaigns

Additional implementation requirements for enterprises using list-based survey campaigns include OMO and Fulfillment steps.

OMO

Implementation of list management aspects of OMO are required. Once these prerequisites are met:

- OMO can be used to create and administer lists for use with surveys.
- Lists can be generated (OMO functionality) from within the Survey Admininstration console.

See Also

See *Oracle Marketing Online Implementation Guide* for more information on implementing OMO. See *Oracle Marketing Online Concepts and Procedures Manual* for more information on working with and generating OMO lists.

Additional Fulfillment implementation and configuration requirements are prerequisite for executing list-based survey campaigns. This includes:

- Implementation and configuration of 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, Fulfillment can take OMO 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.

2.2.5 Requirements for Administering Survey Resources and 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 JSP-format files that display when a respondent takes a survey. Chief among these is a header which appears on each HTML page (each panel in the script appears as a single HTML page). The other resources are an error page that results in the even of a Java stack error when processing the survey, and a final page that appears after the last panel in the survey. For planning purposes, you must determine the requirements for these resources (what they must look like, if they already exist, whether the JSP-format page includes any live JSP elements or is simple HTML, and whether default resources should be designated for each survey administrator or for the site). This information is entered by a Survey administrator in the Survey Admininstration 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 Campaign** subtab. A survey campaign may contain more than one cycle. Cycles can also be created from the **Cycle** subtab of the **Survey Campaign** tab.

Note: The order of steps described here is not arbitrary. Even though survey resources are administered from the fourth (Resources) subtab of the Survey Campaign tab, they must be in place *prior* to creating a survey campaign in the first (Survey Campaign) subtab of the Survey Campaign tab, since the identification of resources is survey campaign-specific and must occur before you can save a new survey campaign.

For planning purposes, the following types of information are required:

Survey Campaign-Level Information:

- Script name
- Survey campaign name
- Survey resources:
  - Defaults from profile (must be established in advance) or campaign-specific?
  - Names for Header, Error, and Final Page survey resources (must be established in advance)
  - Cycle-Level Information
- Number of cycles required
- Cycle names
- Requirement for anonymous flag
- Minimum response percentage

See Also

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for Reporting Survey Campaign Deployment Results

2.2.6 Requirements for 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 non-list-based, all list-related information on the **Create Deployment** page should be left blank.

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.
- Deploy date and time
- Response end date and time
- Enterprise Web server URL and port

List-Specific Information

- List name
- Invitation template name
- Reminder template name
- Maximum responses per person
- Minimum responses for close
- E-mail message subject heading
- Number of reminders

See Also

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for Reporting Survey Campaign Deployment Results

2.2.7 Requirements for Deploying Survey Campaigns

Once one or more deployments have been defined under a cycle for a survey campaign, it can be deployed immediately.

Non-List-Based Scenario

For non-list-based campaigns, the act of deploying (setting a deployment to Active status) enables respondents to take surveys as soon as you press the Deploy button.

List-Based Scenario

For list-based campaigns, 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.
- A Fulfillment Template, associating a specific Master Document and its components.
- Fully configured lists, typically generated in OMO.
- 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 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 OMO 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

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Monitoring Survey Results

Collecting Survey Results in Oracle RDBMS

Requirements for 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 Admininstration console, each individual response can be reviewed on an ongoing basis from the **Response** tab in the Survey Admininstration console. You must drill down to the specific deployment to see responses for that deployment.

See Also

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Collecting Survey Results in Oracle RDBMS

Requirements for 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

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

Requirements for Defining Deployments

Requirements for Deploying Survey Campaigns

Monitoring Survey Results

Requirements for Reporting Survey Campaign Deployment Results

2.2.10 Requirements for Reporting Survey Campaign Deployment Results

You can view four types of reports (Response Summary, List Summary, Question Frequency, or Panel Footprint) by selecting the **Analysis** tab from the Survey Admininstration console. Each has its own required parameters to query from the **Analysis** tab. You must have the appropriate information for each report in order to query the information to display. If no surveys in a deployment have received responses, no information will be available to display in these reports.

Additionally, for each report except the Footprinting report, 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, Release 11i*.

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 inAdministering Concurrent Programs for Survey Execution in *Oracle Scripting Implementation Guide, Release 11i.*

For more information, see Reporting and Analyzing Survey Respondent Data in *Oracle Scripting Implementation Guide, Release 11i*.

See Also

Gathering All Survey Campaign Requirements

Requirements for Creating and Deploying a Survey Questionnaire

Requirements for Determining If Survey Campaign Is List-Based

Requirements for Generating Lists

Requirements for Administering Survey Resources and Survey Campaign and Cycle Details

3 Using Oracle Scripting

Oracle Scripting is a set of tools, each with its own users. End users (users of a script) include survey respondents (for the Survey component) and interaction center agents (for the Scripting Engine component). Users of the Script Author include script developers (trained functional users with some technical knowledge), and Java, PL/SQL, and Oracle Forms developers. Individuals with strong interaction center and campaign planning backgrounds participate in survey campaign requirements definition and survey campaign administration using the Survey Admininstration console.

Using Oracle Scripting involves all these different facets. Some of this information is dispersed throughout different sections of this document. Reference will be made herein to other sections as appropriate.

This topic includes the following topic groups:

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

See Also

Who Are the End Users of Oracle Scripting?

3.1 Using the Script Author

This topic group provides process-oriented, task-based procedures for using the application to perform essential business tasks. Some subtopics fall into several categories. Thus a single subtopic may be referenced under multiple locations in this document.

This topic group includes the following topics:

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working with Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Reusing Commands

Deploying the Script

See Also

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.1 Getting Started with the Script Author

There is no single correct manner in which to create scripts in the Script Author. However, certain information is essential, and a recommended flow of events is described in this section. Other task-oriented processes are described in the remainder of this section, including step-by-step procedures to use the Script Author effectively.

To develop a new script you must obtain script campaign requirements, insert the scripting objects needed to implement the business rules into the Script Author, associate properties for each script object, ensure branches will provide the appropriate flow, define global script attributes based on your requirements, save the script, and deploy it to the applications database. The specific major tasks associated with this process are indicated below.

Steps

1. Obtain script campaign requirements.
2. Start a new script project.
3. Insert appropriate script objects.
4. Define properties for each configurable script object (panels, groups, and blocks).
5. Insert appropriate branching to meet flow objectives.
6. Define global script attributes, such as:
    a. Script properties (Script Name, Comments, Script Language, Footprinting and Answer Collection
    b. Global script pre- and post-actions
    c. Script information panel
    d. Shortcut buttons
    e. Script Disconnect button
7. Check the syntax of the script.
8. Save the script.
9. Deploy the script to the Oracle Applications database.

See Also

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.2 Obtaining Script Requirements Before Creating a Script

From a planning perspective, a script is typically part of a campaign that must be executed. A campaign can be defined as a focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose. This concept is typical of interaction centers and is supported from a software perspective in a variety of software products in the CRM portion of the eBusiness Suite.

Prior to inserting a single panel on the canvas, 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 campaign 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 campaign 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 campaign using Script Author objects and custom code.

For more information, see Planning Scripting Engine Projects.

3.1.3 Obtaining Script Requirements Before Creating a Script

From a planning perspective, a script is typically part of a campaign that must be executed. A **campaign** can be defined as a focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose. This concept is typical of interaction centers and is supported from a software perspective in a variety of software products in the CRM portion of the eBusiness Suite.

Prior to inserting a single panel on the canvas, 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 campaign 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 campaign 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 campaign using Script Author objects and custom code.

For more information, see the *Oracle Scripting Implementation Guide* appendix, Scripting Project Scoping and Discovery.

See Also

Getting Started with the Script Author

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4 Defining Global Script Attributes

Global script attributes are attributes that affect the entire script. These include the script's global properties (including the script name recognized by the database), as well as aspects that appear during runtime in the Scripting Engine: the presence of a script information panel, shortcut buttons, and enabling the Disconnect button. You can perform the following tasks:

Designating Global Script Properties

Defining Global Script Pre- and Post-Actions

Defining the Script Information Panel

Defining Shortcut Buttons

Programming the Script Disconnect Button

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.1 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 the 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
- Setting Footprinting
- Setting Answer Collection for the Script

See Also

Defining Global Script Pre- and Post-Actions

Defining the Script Information Panel

Defining Shortcut Buttons

Programming the Script Disconnect Button

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.1.1 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.

Caution: 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.

Prerequisites

Create or open a script. You must have a script open to designate the script name.

Steps

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

   The Properties dialog box appears.

2. In the Script tree, select **Properties**.
3. In the Name field, type the name of the script.

   Do not include the percent character, single quotes, or double quotes in this name.

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

See Also

Defining Script Comments

Defining Script Language

Setting Footprinting

Setting Answer Collection for the Script

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.1.2 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. Choose **File > Script Properties** or double-click on the canvas away from any objects.

   The Properties dialog box appears.

2. In the Script tree, select **Properties**.
3. In the Comments field, type the desired comment.
4. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Defining the Script Name

Defining Script Language

Setting Footprinting

Setting Answer Collection for the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.1.3 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. Choose **File > Script Properties** or double-click on the canvas away from any objects.

   The Properties dialog box appears.

2. In the Script tree, select **Properties**.

3. In the Script language field, enter the appropriate language to indicate in what language the script is written.

   Any uni-directional language used in FND_LANGUAGES is supported.

See Also

Defining the Script Name

Defining Script Comments

Setting Footprinting

Setting Answer Collection for the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.1.4 Setting Footprinting Use this procedure to set footprinting for the appropriate Oracle Scripting runtime component. This is accomplished on a per-script basis from the Script Author. 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.

Prerequisites

Create or open a script. You must have a script open to set footprinting.

Steps

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

   The Properties dialog box appears.

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

See Also

Defining the Script Name

Defining Script Comments

Defining Script Language

Setting Answer Collection for the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.1.5 Setting Answer Collection for the Script Use this procedure to have Oracle Scripting record the answers to the script panels enabled during an agent interaction. These are stored in the Scripting schema of the Oracle Applications database.

Prerequisites

Create or open a script. You must have a script open to set Answer Collection.

Steps

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

   The Properties dialog box appears.

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

See Also

Defining the Script Name

Defining Script Comments

Defining Script Language

Setting Footprinting

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.2 Defining Global Script Pre- and Post-Actions

Use this procedure to define an action to take place before the first script panel appears (a pre-action) or after the last panel is executed in a script (a post-action).

Prerequisites

Create or open a script. You must have a script open to designate global pre- and post-actions.

Steps

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

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

   The Properties dialog box appears. These are the properties for the global script.

2. In the Script tree, expand **Actions** and then select **Pre Actions** or **Post Actions**.

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

   The Command dialog box appears.

4. Define a command for the action. (See Defining Commands.)

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

See Also

Designating Global Script Properties

Defining the Script Information Panel

Defining Shortcut Buttons

Programming the Script Disconnect Button

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.3 Defining the Script Information Panel

Use this procedure to setup the script information panel for the Scripting Engine. The information panel can contain up to nine data elements (text or a timer). The information panel displays in three rows, each able to display a data element and label at 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 static or dynamic information. This panel 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 panel, populating the data elements in the panel. Using Scripting APIs, timers can be started and stopped, and data in the script information panel can be updated by explicit command.

Prerequisites

Create or open a script. You must have a script open to define script information panel elements.

Steps

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

   The Properties dialog box appears.

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

The Oracle Scripting API requires this ID.

7. In the Label field, enter the label that appears in the script information panel at runtime.
8. In the Command field, click **Edit**.

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

See Also

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.4 Defining Shortcut Buttons

Use this procedure to define a button in the toolbar of the Scripting Engine interface. Clicking this button in runtime will jump the interaction center agent to the group in the script with the appropriate shortcut property.

Prerequisites

Insert a group.

Steps

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

   The Properties dialog box appears.
3. In the Group tree, select **Shortcut**.
4. In the Shortcut pane, type the name of the shortcut.

5. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box for the group.
6. Choose **File > Script Properties**.

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

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

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

See Also

Designating Global Script Properties

Defining Global Script Pre- and Post-Actions

Defining the Script Information Panel

Programming the Script Disconnect Button

Programming the Script Disconnect Button

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.4.5 Programming the Script Disconnect Button

Use this procedure to program the Disconnect button in the Scripting Engine interface. Selecting the Disconnect button at runtime will route 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 What Are the 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.

Steps

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

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

   The Shortcut pane appears.
4. In the Shortcut field in the Shortcut pane, 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 dialog box for the group.

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

See Also

Designating Global Script Properties

Defining Global Script Pre- and Post-Actions

Defining the Script Information Panel

Defining Shortcut Buttons

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.5 Working with Script Files

You can perform the following tasks:

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.5.1 Starting a New Script

Use this procedure to create a new script.

Prerequisites

None

Steps

- Choose **File** > **New**, or click the **New Script** icon in the toolbar.

  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 Start node appears on the canvas.

See Also

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.2 Opening an Existing Script from the File System

Use this procedure to open a saved 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.

Prerequisites

None

Steps

1. Choose **File** > **Open** or click the **Open a Script** icon in the toolbar.

   The **Open a Script** icon is the second icon from the left of the toolbar, located immediately below the menus in the Script Author interface.

   The **Script Chooser** window for the Script Author will open.

2. Click the **File System** tab from the **Script Chooser** window, if not already selected.

3. From the **Location** pull-down menu, select the appropriate local volume.

4. From the **Files** list, locate the script (*.SCRIPT file) that you want to open and then click **Open**.

   The script objects appear 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**).

See Also
Starting a New Script

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.3 Opening an Existing Script from the Applications Database
Use this procedure to open a saved 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.

Prerequisites
You must have appropriate database connection information, including a username with access to the IES_DEPLOYED_SCRIPTS table in the applications database.

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

   The **Script Chooser** window for the Script Author will open.
2. Click the **Database** tab from the **Script Chooser** window, if not already selected.
3. Click **Edit** (located next to the **Connection** field).

   The **Edit Profile** window appears. Information entered into this window defines your database connection information.
4. Enter database connection information for your environment.
   a. Optionally, in the **Connection Name** field, enter a name for this connection.

      This information is used by the Script Author only. Any characters may be used as part of this name. It is not necessary to name a connection.

   b. In the Host Name field, type the name of the database host (including domain).

      For example, type **database.vision.com**.

   c. In the Port Number field, enter the TNS port of the database.

   d. In the SID field, enter the database global name or SID.

   e. In the User ID field, enter an appropriate database User ID.

      This user must have privileges to access the IES_DEPLOYED_SCRIPTS table of the applications database (for example, the apps user).

   f. Once you have entered all appropriate connection information, click **OK**.

      The Edit Profile window will close. The Connection field will be populated with the connection name and User ID entered in the previous step.

5. In the Password field, enter the password for the configured User ID.
6. Optionally, you can select from the **Published** or **Deployed** radio buttons to view different scripts.
7. Locate the script (*.SCRIPT file) that you want to open in the Scripts list and then click **Open**.

   The script objects appear 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**).

See Also

Starting a New Script

Opening an Existing Script from the File System

Saving a Script to the File System

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 the Script Author

3.1.5.4 Saving a Script to the File System

Use this procedure to save a script to the file system of a local volume or mounted storage medium. You can save a script using its current name and location, or save a copy of the script using a different name and location. This name must not have any special characters, such as spaces, slashes, or backslashes, which are not permitted in the file system of some operating systems.

Note: In order for the Script Author to be able to recognize a script file, it must end with a .SCRIPT or a .SCR file extension.

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 at runtime. Therefore, you may wish to use the same name in the script properties that you identify in this step for the file name.

Note: In the title bar of the Script Author, the asterisk to the right of the script name indicates that changes have been made to the script that have not yet been saved.

Prerequisites

None

Steps

1. Do one of the following:

   - To save the script using its current name and location, choose **File** > **Save** or click the **Save Script** icon in the toolbar.

     The **Save Script** icon is the third icon from the left of the toolbar, located immediately below the menus in the Script Author interface.

- To save a copy of the script using a different name and location, choose **File** > **Save As**.
- Ensure the script name ends with a .SCRIPT or .SCR file extension.

See Also

Defining the Script Name

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 the Script Author

3.1.5.5 Reversing All Changes Since the Last Save Command

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

Prerequisites

None

Steps

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

See Also

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

3.1.5.6 Importing a Script

Use this procedure to import a previously existing script into the current script.

The import command imports all objects in an existing script. If you wish to import only a portion of a script, you can (1) export that portion from the existing script and then import it as described below, or (2) import the entire script as described below and discard the unwanted portions.

Caution: Every Script Author script is essentially a customized product. It is the script developer's responsibility to ensure **imported** scripts will function in the target environment. If a script 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.

Prerequisites

None

Steps

1. Choose **File > Import**.

   The Open dialog box appears.

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

   The imported script appears on the canvas as a group.

See Also

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.7 Exporting a Script Group as a Separate Script File

Use this procedure to export a group as a separate script file. Exported groups can then be imported and used in other scripts, providing for modularity and code reuse.

Prerequisites

None

The import command imports all objects in an existing script. If you wish to import only a portion of a script, you can (1) export that portion from the existing script and then import it as described below, or (2) import the entire script as described below and discard the unwanted portions.

Caution: Every Script Author script is essentially a customized product. It is the script developer's responsibility to ensure exported groups will function in the target environment. If a script component you export 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.

Steps

1. On the canvas, select a group.
1. Choose **File > Export**.

   The Save dialog box appears.
2. Select the destination.
3. Type the file name and then click **OK**.

See Also

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.8 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.

See Also

Fitting a Script Layout in the Canvas

Drilling Up or Down to a Related Script

Aligning Objects

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.9 Closing a Script

Use this procedure to close a 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

Saving a Script to the File System

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.10 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 tool palette, that object type remains "stuck" (selected) until you explicitly select another object or branch type from the tool palette. 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 tool palette, each 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, that option is then "stuck" until another explicit selection is made 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 is to designate properties for the current object.

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. To disable sticky mode, when this option is selected, click to clear this option.

   The **File** menu closes, and sticky mode is toggled on. 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.

   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 from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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 the Script Author

3.1.5.11 Exiting the Script Author

Use this procedure to exit the Script Author application. 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**.

   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

Saving a Script to the File System

Starting a New Script

Opening an Existing Script from the File System

Opening an Existing Script from the Applications Database

Saving a Script to the File System

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

3.1.6 Working with the Tool Palette

You can perform the following tasks:

Inserting an Object

Inserting a Branch

See Also

Selecting Objects

Deselecting Objects

Moving Objects

Moving Branches

Deleting Panels, Groups, Blocks and Termination Nodes

Deleting Branches

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.6.1 Inserting an Object

Script Author has three configurable objects (panels, blocks and groups) and two non-configurable objects (the Start and Termination nodes). These objects must be attached with branches to direct the flow of the script.

- If you want to display information (such as text, an image, a hyperlink, a variable, or a Java bean) to the script viewer at runtime or accept information entered by the script viewer, 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 Dialog Box to Pop Up at Object Creation
- Inserting a Panel
- Inserting a Group
- Inserting a Block
- Inserting a Termination Node

3.1.6.1.1 Setting Properties Dialog Box to Pop Up at Object Creation Use this procedure to set up Script Author to automatically display the Properties dialog box 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 What Are the Minimum Requirements for Any Graph?

Prerequisites

None

Steps

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

See Also

Inserting a Panel

Inserting a Group

Inserting a Block

Inserting a Termination Node

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.6.1.2 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 What Is a Script? > 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 tool palette, 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 Dialog Box to Pop Up at Object Creation

Inserting a Panel

Inserting a Group

Inserting a Block

Termination Nodes

Deleting Panels, Groups, Blocks and Termination Nodes

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.6.2 Inserting a Branch

Use this procedure to insert a branch onto the canvas to connect objects. Branches direct the flow of the script based on the branch type and properties. For more information on branches, see Defining Branches.

Prerequisites

- Insert an object from which to begin branching (or use Start node).
- Insert a destination object.

Steps

1. In the tool palette, click the appropriate branch type tool.

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

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.

See Also

Defining Branches

Inserting a Panel

Inserting a Group

Inserting a Block

Inserting a Termination Node

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.7 Viewing Objects on the Canvas

You can perform the following tasks:

Fitting a Script Layout in the Canvas

Drilling Up or Down to a Related Script

Aligning Objects

See Also

Printing a Graph on the Script Author Canvas

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.7.1 Fitting a Script Layout in the Canvas

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

Prerequisites

None

Steps

Do one of the following:

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

See Also

Drilling Up or Down to a Related Script

Aligning Objects

Printing a Graph on the Script Author Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.7.2 Drilling Down Into or Up From a Group or Block

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

Prerequisites

None

Steps

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

See Also

Fitting a Script Layout in the Canvas

Aligning Objects

Printing a Graph on the Script Author Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.7.3 Aligning Objects

Use this procedure to align objects vertically or horizontally.

Prerequisites

None

Steps

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

See Also

Fitting a Script Layout in the Canvas

Drilling Up or Down to a Related Script

Printing a Graph on the Script Author Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.8 Working with Objects on the Canvas

You can perform the following tasks:

Selecting Objects

Deselecting Objects

Moving Objects

Moving Branches

Deleting Panels, Groups, Blocks and Termination Nodes

Deleting Branches

See Also

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

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

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

See Also

Deselecting Objects

Moving Objects

Moving Branches

Deleting Panels, Groups, Blocks and Termination Nodes

Deleting Branches

Fitting a Script Layout in the Canvas

Drilling Up or Down to a Related Script

Aligning Objects

Printing a Graph on the Script Author Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.8.2 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, Shift-click the object. You can also Control-click the object.
- To deselect all selected objects, click on the canvas away from any objects.

See Also

Selecting Objects

Moving Objects

Moving Branches

Deleting Panels, Groups, Blocks and Termination Nodes

Deleting Branches

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.8.3 Moving Objects

Use this procedure to move panels, groups, and blocks on the canvas.

Prerequisites

None

Steps

1. In the tool palette, 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.

See Also

Selecting Objects

Deselecting Objects

Moving Branches

Deleting Panels, Groups, Blocks and Termination Nodes

Deleting Branches

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.8.4 Moving Branches

Use this procedure to change the destination object for a branch.

Prerequisites

None

Steps

1. In the tool palette, 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. To retain the object, use the Edit command to copy the object and paste it on the canvas.

3. Choose **Edit** > **Delete**.

   The object that was previously the destination of the branch is deleted.

4. Select 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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.8.5 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 tool palette, 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

Moving Branches

Deleting Branches

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.8.6 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 tool palette, 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

Moving Branches

Deleting Panels, Groups, Blocks and Termination Nodes

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9 Defining Panels

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

You can perform the following tasks:

Inserting a Panel

Defining Panel Properties

Defining Panel Answer Controls

Defining Panel Text

See Also

Defining Panel Pre- and Post-Actions

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.1 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, displaying panel text, any answers (nodes) that have been defined for the panel, and associated labels for those answers. For more information on panels, see Defining Panels.

Prerequisites

None

Steps

1. In the tool palette, 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 dialog box for the object appears.

Caution: Ensure you define an answer for each panel created, or the script will not pass a syntax check. In runtime, every panel requires end user interaction (response to an answer definition).

See Also

Setting Properties Dialog Box to Pop Up at Object Creation

Inserting a Group

Inserting a Block

Inserting a Termination Node

Defining Panel Properties

Defining the Panel Name

Defining the Panel Comments

Defining the Panel Label

Defining Vertical or Horizontal Answer Control Layout

Substituting a Java Bean for a Panel

Defining Panel Pre- and Post-Actions

Defining Panel Answer Controls

Defining Panel Text

What Are the Minimum Requirements for Any Graph?

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.2 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
- Substituting a Java Bean for a Panel

Prerequisites

Insert a panel onto the canvas.

3.1.9.2.1 Defining the Panel Name Use this procedure to define the name of a panel. The panel name is how the panel is identified in Script Author. Note that the panel name is distinct from the panel label, which is displayed in the Progress Area of the Scripting Engine at runtime.

Prerequisites

Insert a panel onto the canvas.

Steps

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

The Properties dialog box appears.

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

See Also

Inserting a Panel

Defining Panel Comments

Defining the Panel Label

Defining Vertical or Horizontal Answer Control Layout

Substituting a Java Bean for a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.2.2 Defining Panel Comments Use this procedure to define a comment for a panel. There is no maximum character limit to a comment and no character restrictions.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

See Also

Defining the Panel Name

Defining the Panel Label

Defining Vertical or Horizontal Answer Control Layout

Substituting a Java Bean for a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.2.3 Defining the Panel Label Use this procedure to define the label for a panel. The panel label is the value used to identify specific panels in the Progress Area of the Scripting Engine at runtime. Only panels that have been visited by the end user in the flow will be listed (by panel label) in the Progress Area. Note that the value the Script Author uses to reference panels is the panel name, not the panel label.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

See Also

Inserting a Panel

Defining the Panel Name

Defining Panel Comments

Defining Vertical or Horizontal Answer Control Layout

Substituting a Java Bean for a Panel

Packaging Java Bean Code Into a JAR File

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.2.4 Substituting a Java Bean for a Panel Use this procedure to substitute a Java bean for a panel.

Caution: 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a panel.

   The Properties dialog box appears.
3. In the Panel tree, select **Properties**.
4. In the Properties pane, select **Replace with a Java Bean**.

5. In the Bean Name field, type the full path and name of the Java bean (for example, mybeans.foobean).
6. In the Jar File Name field, type the full path and name of the jar file for the Java bean.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Inserting a Panel

Defining the Panel Name

Defining Panel Comments

Defining the Panel Label

Defining Vertical or Horizontal Answer Control Layout

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.2.5 Packaging Java Bean 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. In order for a script with a custom Java Bean to supply that code to the Scripting Engine GUI at runtime, the code for the Java Bean must first be packaged into a .JAR file. Due to a limitation in JDK 1.1.8, there are only two supported methods of packaging the JAR file appropriately. Either of these methods described below can be used to package a .JAR file appropriately.

Caution: While an entire panel can be substituted, substituting a Java bean for *an answer* (or panel node) *within a panel* is no longer supported in Oracle Scripting as of release 11.5.6 and subsequent releases, due to the new WYSIWYG editing feature of the Script Author.

Prerequisites

- As with all custom code, the Java bean 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 against the Java Development Kit (JDK) 1.1.8.

Steps

- Use the "jar" utility from JDK 1.1.8 and specify no compression:

  ```
  jar -cf0 TestBean.jar ...).
  ```

- Create the custom JAR file in the standard .ZIP format (with any PC-standard compression utility) with or without compression, and then simply rename the file extension from .ZIP to .JAR.

3.1.9.3 Defining Panel Text

You can perform the following tasks:

- Entering Panel Text
- Inserting a Hyperlink
- Inserting an Embedded Value
- Inserting an Image
- Exporting Panel Text to an HTML File
- Importing an HTML File into the Panel Layout Editor

See Also

Inserting a Panel

Defining the Panel Name

Defining Panel Comments

Defining the Panel Label

Defining Panel Pre- and Post-Actions

Substituting a Java Bean for a Panel

3.1.9.3.1 Entering Panel Text Use this procedure to define the text that displays in the script panel at runtime.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

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

See Also

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Layout Editor

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

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

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

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

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

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

See Also

Entering 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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.3.3 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. In the tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, select a panel.
3. From the menu, choose **Tools > Panel Layout Editor**.

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

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

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

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

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

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

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

See Also

Entering Panel Text

Inserting a Hyperlink

Inserting an Image

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Layout Editor

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

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

Prerequisites

Insert a panel onto the canvas.

Steps

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

2. On the canvas, select a panel.

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

The panel layout editor appears.

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

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

   The image appears at the insertion point.
8. If you want to save the text to the panel, then choose **File > Save** from the menu.
9. If you want to save the text to an HTML file, then choose **File > Export** from the menu.

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

See Also

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Exporting Panel Text to an HTML File

Importing an HTML File into the Panel Layout Editor

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.3.5 Exporting Panel Text to an HTML File Use this procedure to save panel text as an HTML file.

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

Prerequisites

Insert a panel onto the canvas.

Steps

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

   The panel layout editor appears.
4. If you want to save the text as an HTML file using its current name and location, then choose **File > Export**.
5. If you want to save the text as an HTML file using a different name and location, then choose **File > Export As**.

   If you choose the Export As command, or if the script has never been exported, then the Save dialog box appears.
6. Specify the location and filename.
7. Click **Save**.

See Also

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Importing an HTML File into the Panel Layout Editor

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.9.3.6 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 text previously in the panel layout editor.

Prerequisites

Insert a panel onto the canvas.

Steps

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

   The panel layout editor appears.
4. Choose **File > Import**.

   The Open dialog box appears.
5. Locate the file that you want to import.
6. Click the file and then click **Open**.

   The HTML appears in the panel layout editor.

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

Entering Panel Text

Inserting a Hyperlink

Inserting an Embedded Value

Inserting an Image

Exporting Panel Text to an HTML File

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.10 Defining Groups

You can perform the following tasks:

Inserting a Group

Importing a Saved Script as a Group

Defining the Group Name

Defining Shortcuts

See Also

Defining Group Pre- and Post-Actions

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.10.1 Inserting a Group

Use this procedure to insert a group onto the canvas. For more information on groups, see Defining Groups.

Caution: 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 tool palette, 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** button on the canvas toolbar is made active. If the Popup on Blob Creation option is selected, then the Properties dialog box for the object appears.

See Also

Setting Properties Dialog Box to Pop Up at Object Creation

Inserting a Panel

Inserting a Block

Inserting a Termination Node

Drilling Up or Down to a Related Script

Importing a Saved Script as a Group

Defining the Group Name

Defining Shortcuts

Defining Group Pre- and Post-Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.10.2 Importing a Saved Script as a Group

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

Prerequisites

None

Steps

1. Choose **File > Import**.

   The Open dialog box appears.

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

   The script appears on the canvas as a group.

See Also

Inserting a Group

Defining the Group Name

Defining Shortcuts

Defining Group Pre- and Post-Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.10.3 Defining the Group Name

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

Prerequisites

Insert a group onto the canvas.

Steps

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

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

See Also

Defining Group Pre- and Post-Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.10.4 Defining Shortcuts

A Shortcut is the property of a Group in the Script Author. Using Java commands, the flow of a script in runtime can jump to a "target" group containing a Shortcut of a specified name. The group contains the panels that are the destination that result when clicking the Shortcut button. Three typical uses of a Shortcut are to associate a target for a Shortcut button in the Shortcut button bar (for the agent interface only), to enable the Disconnect button (for the agent interface only), and 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).

Use this procedure to define a group as the target of a jump.

Prerequisites

Insert a group.

Steps

1. Using the Script Author, open a new or existing script.
2. In the tool palette, click the **Toggle Select Mode** tool.
3. On the canvas, double-click a group.

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

   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 Terminal node.

3.1.11 Defining Blocks

Using blocks, you can integrate data sources into your script. With blocks you can query, update, or insert information in database tables, as well as associate commands. The block gives a clear visual indicator to the script developer that data source integration is occurring at that point in the script.

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

See Also

Defining Block Pre- and Post-Actions

Defining Panel Answer Controls

Defining Actions

Defining Commands

Defining Shortcuts

3.1.11.1 Inserting a Block

Use this procedure to insert a block onto the canvas. For more information on blocks, see Defining Blocks.

Caution: 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 objects as appropriate.

Prerequisites

None

Steps

1. In the tool palette, 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.

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

See Also

Defining the Block Name

Defining Block Pre- and Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Defining a Database Update Block

Defining Database Connections for a Block

Defining Database Tables for a Block

Defining Join Conditions for a Block

Defining Data Constraints for an Insert or Update Block

Defining Query Constraints for a Query Block

Defining Query Columns for a Query Block

Defining Panels within a Block

3.1.11.2 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a block.

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

See Also

Defining Block Pre- and Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Defining a Database Update Block

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.3 Defining a Database Query Block

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

Prerequisites

Insert a block onto the canvas. In addition, you need the following information:

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

Steps

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

   The Properties dialog box appears.
3. In the Block tree, select **Type**.
4. In the Type pane, select **Query Block**.
5. In the Block tree, select **Object Dictionary**.
6. In the Connection/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 Constraints tab,

   a. If you want to improve network performance during the query, then type the number of rows to be returned at a time during retrieval of the query result in the Prefetch Value field.

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

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

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

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

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

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

See Also

Defining the Block Name

Defining Block Pre- and Post-Actions

Defining a Database Insert Block

Defining a Database Update Block

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.4 Defining a Database Insert Block

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

Prerequisites

Insert a block onto the canvas. In addition, you need the names of the database tables into which you want to insert a record.

Steps

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

   The Properties dialog box appears.

3. In the Block tree, select **Type**.
4. In the Type pane, select **Insert Block**.
5. In the Block tree, select **Object dictionary**.
6. In the Connection/Table tab, do the following:
   a. Define how Oracle Scripting connects to the database at runtime. (See Defining Database Connections for a Block.)
   b. List the database tables into which you want to insert information. (See Defining Database Tables for a Block.)
7. If you want to insert data into more than one database table, then, in the Join Condition tab, indicate the relationship between related tables. (See Defining Join Conditions for a Block.)

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

8. If you want to insert a panel answer into a database table, then do the following:
   a. Drill down into the block
   b. Insert a panel in the block.
   c. Insert a Termination node onto the canvas.
   a. Draw branches between the objects to indicate the flow of the script.
   b. Define an answer. (See Defining Panel Answer Controls.)
   c. From the Panel Properties dialog, select the answer definition and click **Edit**.

      The Answer Entry dialog box appears.
   d. Click **Edit Data Dictionary**.

      The Edit Data Dictionary dialog appears. The default data dictionary name is *answername*DataDictionary[number].
   e. Click the **Table Info** tab.

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

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

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

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

---

i. Click **OK** to exit the Answer Entry dialog box.

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

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

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

      The Data Constraint dialog appears.

   b. In the Command area, click Edit.

      The Command dialog box appears.

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

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

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

---

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

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

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

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

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

See Also

Defining the Block Name

Defining Block Pre- and Post-Actions

Defining a Database Query Block

Defining a Database Update Block

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.5 Defining a Database Update Block

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

Prerequisites

Insert a block onto the canvas. In addition, you need the following information:

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

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

Steps

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

   The Properties dialog box appears.
3. In the Block tree, select **Type**.
4. In the Type pane, select **Update Block**.

5. In the Block tree, select **Object dictionary**.
6. In the Connection/Table tab, do the following:
   a. Define how Oracle Scripting connects to the database at runtime. (See Defining Database Connections for a Block.)
   b. List the database tables that contain the information that you want to update. (See Defining Database Tables for a Block.)
7. If you want to update data in more than one database table, then, in the Join Condition tab, indicate the relationship between related tables. (See Defining Join Conditions for a Block.)

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

8. If you want to use a panel answer to update a database table, then do the following:
   a. Drill down into the block
   b. Insert a panel in the block.
   c. Insert a Termination node onto the canvas.
   d. Draw branches between the objects to indicate the flow of the script.
   e. Define an answer. (See Defining Panel Answer Controls.)
   f. From the Panel Properties dialog, select the answer definition and click **Edit**.

      The Answer Entry dialog box appears.
   g. Click **Edit Data Dictionary**.

      The Edit Data Dictionary dialog appears. The default data dictionary name is *answername*DataDictionary[number].
   h. Click the **Table Info** tab.
   i. In the Table field, type the name of the table to be updated.
   j. In the Column field, type the name of the column to be updated.
   k. Click **OK** to exit the Data Dictionary dialog box.

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

- l. Click **OK** to exit the Answer Entry dialog box.
- m. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

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

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

     The Data Constraint dialog appears.
   - b. In the Command area, click Edit.

     The Command dialog box appears.
   - c. Define a command that obtains or derives the value used to update the table. (See Defining Commands.)
   - d. In the Name field, type the name of the data constraint.

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

   - e. In the Table field, type the name of the table to be updated.
   - f. In the Column field, type the name of the column to be updated.
   - g. Click **OK** to exit the Data Constraints dialog box.
   - h. If you want to add another data constraint, then repeat steps a through g.

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

See Also

Defining the Block Name

Defining Block Pre- and Post-Actions

Defining a Database Query Block

Defining a Database Insert Block

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.6 Defining Database Connections for a Block

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

Prerequisites

Insert a block onto the canvas.

Steps

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

   The Properties dialog box appears.
3. In the Block tree, select **Object dictionary**.
4. If you want to define new connection parameters, then, in the Connection/Tables tab, do the following:
   a. If necessary, clear the **Reuse an existing connection** box.
   b. Type the connection parameters. (See Database Connection Guidelines.)

   Note: When you save your work in the Properties dialog box, the connection parameters are available in the Existing Connections list for this script and can be reused in another block or when deploying the script.
5. If you want to reuse an existing connection, then, in the Connections/Tables tab, do the following:
   a. Select the **Reuse an existing connection** box.

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

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

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.7 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a block.

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

   The Table dialog box appears.

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

> **Note:** List master tables first and then tables related by foreign key.

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

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.8 Defining Join Conditions for a Block

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

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

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

Prerequisites

Insert a block 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a block.

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

   The Join Condition Dialog dialog box appears.
5. In the Master PK Table tab, do the following:
   a. In the Master Table field, type the name of the table.
   b. Click **Add Master PK**.

      The Column Entry dialog box appears.

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

   c. Type the name of the column that is the primary key of the master table and then click **OK** to exit the Column Entry dialog box.
   d. If you want to add another primary key, then repeat steps b through c.
6. In the Detail PK Table tab, do the following:
   a. In the Detail Table field, type the name of the related table.
   b. Click **Add Detail PK**.

   The Column Entry dialog box appears.

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

---

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

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

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

   a. Click **Add Detail FK**.

   The Column Entry dialog box appears.

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

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

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

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

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.9 Defining Data Constraints for an Insert or Update Block

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

Prerequisites

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

Steps

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

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

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

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.10 Defining Query Constraints for a Query Block

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

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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a query block.

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

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

   Note: Conditions based on user input require a panel in the block. The *value* is an answer name in the panel. See Defining Panels within a Block.

6. Click **OK** to exit the Query Constraint Input dialog box.
7. If you want to add another query constraint, then repeat steps 4 through 6.

   Note: Insert logical operators (such as, NOT, AND, and OR) at the end of the query constraint. If no logical operator is used, then AND is assumed. Use parentheses at the beginning or end of a query constraint to override the rules of precedence.

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

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.11 Defining Query Columns for a Query Block

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

Prerequisites

Insert a block 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a query block.

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

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

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

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

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.11.12 Defining Panels Within a Block

Use this procedure to add a panel to a block. A block serves as a method to add a database function such as select (query), insert or update statement, or to add a command such as an API. It also serves as a container, creating a subgraph that may contain panels, groups or other blocks.

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.

Prerequisites

Insert a block onto the canvas.

Steps

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

   The graph for the block appears.
3. Define the graph, adding panels (or other objects, including.

See Also

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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12 Defining Branches

Branches directly control the flow of a script in runtime. A branch must always begin 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.

You can perform the following tasks:

Setting Properties Dialog Box 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

Working with Branches on the Canvas

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.1 Setting Properties Dialog Box to Pop Up at Branch Creation

Use this procedure to set up Script Author to automatically display the Properties dialog box when you insert the branch onto the canvas.

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

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.2 Defining a Default Branch

Use this procedure to navigate the script 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 should be reordered to designate the default branch as last). Otherwise, other branches would not be evaluated.

- 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 should be reordered to designate the default branch as last). Otherwise, other branches would not be evaluated.
- 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 must be used from any object on the canvas that leads to a Termination node.
- 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."

Prerequisites

Two objects must exist on the canvas in order to connect them using the default branch.

Steps

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

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

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.

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Defining a Distinct Branch

Defining a Conditional Branch

Defining an Indeterminate Branch

Defining the Branch Name

Reordering Branches

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.3 Defining a Distinct Branch

Use this procedure to navigate the script to a destination object based on an answer selected in the source panel. Distinct branches must originate from a panel object only. The distinct branch is the only branch type with this restriction.

> **Caution:** Inserting a distinct branch requires a value to be defined for that branch, as described below. If omitted, the script will not be syntactically correct.

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.
- Insert a distinct branch from the source panel to any destination object.

- Define an answer control for the script panel.

Steps

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

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

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

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

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

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Defining a Default Branch

Defining a Conditional Branch

Defining an Indeterminate Branch

Defining the Branch Name

Reordering Branches

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.4 Defining a Conditional Branch

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

Caution: Inserting a distinct branch requires a value to be defined for that branch, as described below. If omitted, the script will not be syntactically correct.

Note: The custom code referenced by the command defined for the conditional branch is not verified to exist in order to check the syntax of the script. 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 deployed to the server and any configuration files adjusted accordingly so the custom code can be referenced. 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a conditional branch.

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

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

The command must return true or false.

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

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Defining a Default Branch

Defining a Distinct Branch

Defining an Indeterminate Branch

Defining the Branch Name

Reordering Branches

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.5 Defining an Indeterminate Branch

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

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 to be evaluated to be connected wtih branches, and (2) the requirement for the graph containing the indeterminate branch to include a Termination node. This second requirement still applies if the destination object (the destination that the flow will jump to at runtime) is on the same graph as the indeterminate branch.

Caution: Inserting an indeterminate branch requires an expression to be defined for that branch, as described below. If omitted, the script will not be syntactically correct.

Note: The custom code referenced by the command defined for the indeterminate branch is not verified to exist in order to check the syntax of the script. 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 deployed to the server and any configuration files adjusted accordingly so the custom code can be referenced. For more information, contact your systems administrator or refer to the *Oracle Scripting Implementation Guide*.

Prerequisites

- Any panel, group or block object must exist on the canvas in order to use the indeterminate branch.
- Write the code for the expression (typically a Java method).
- Insert an indeterminate branch.

Steps

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

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

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

   The command must return the name of:

   a. 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

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

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Defining a Default Branch

Defining a Distinct Branch

Defining a Conditional Branch

Defining the Branch Name

Reordering Branches

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.6 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.

Prerequisites

Insert a branch onto the canvas.

Steps

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

   The Properties dialog box appears.

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

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Defining a Default Branch

Defining a Distinct Branch

Defining a Conditional Branch

Defining an Indeterminate Branch

Reordering Branches

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.7 Reordering Branches

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

Prerequisites

- Insert a panel onto the canvas.
- Insert branches onto the canvas.

Steps

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

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

4. In the Branches pane, select a branch.
5. If you want the branch to be evaluated at runtime earlier than any other branch in the list, then click **Move Up**.
6. If you want the branch to be evaluated at runtime later than any other branch in the list, then click **Move Down**.
7. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Setting Properties Dialog Box 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

Working with Branches on the Canvas

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.8 Working with Branches Visually on the Canvas

Once a branch has been defined on the canvas, it can be modified visually to accommodate the objects placed on the canvas and create a clear picture of the intended flow of the script. 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

3.1.12.8.1 Inserting a Straight Branch Use this procedure to insert a straight line branch with no corners.

Prerequisites

To insert a branch, you must have at least two object on the canvas.

Steps

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

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

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.

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Inserting a Branch with Corners

Adding a Corner to an Existing Branch

Moving a Corner of a Branch

Deleting a Corner from a Branch

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.8.2 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 tool palette, click a branch tool.

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

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.

See Also

Setting Properties Dialog Box to Pop Up at Branch Creation

Inserting a Straight Branch

Adding a Corner to an Existing Branch

Moving a Corner of a Branch

Deleting a Corner from a Branch

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.8.3 Adding a Corner to an Existing Branch Use this procedure to add a corner to an existing branch.

Prerequisites

None

Steps

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

2. On the canvas, 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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.8.4 Moving a Corner of a Branch Use this procedure to move a corner of a branch.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. 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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.12.8.5 Deleting a Corner from a Branch

Use this procedure to delete a corner from a branch.

Prerequisites

None

Steps

1. In the tool palette, click the **Toggle Select Mode** tool.
2. 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

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.13 Controlling Script Flow

Script flow is controlled by several aspects. The first and most obvious method is defining branches. However, script flow is also controlled by other aspects, such as the following, each with its own section:

Defining Panel Answer Controls

Defining Actions

Defining Blocks

Defining Commands

Defining Shortcuts

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Defining Panel Answer Controls

Working With Answers

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14 Defining Panel Answer Controls

Answer controls are the user interface controls that display in a script at runtime. Every panel requires at least one answer definition or node to be defined. Each node may contain a single user interface type and may have text associated with the answer control. The end user (interaction center agent or survey respondent) must interact with an answer control for each panel, whether the panel contains a single node or multiple nodes. If an answer definition includes validation, that panel at runtime requires the end user to interact with that answer control following the parameters dictated by the code behind the validation command.

You can perform the following tasks:

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Defining a Password Answer Control in a Panel

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.1 Defining a Text Field Answer Control in a Panel

Use this procedure to define a text field answer 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 answer node.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

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

   The Answer Entry Dialog dialog box appears.

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

> **Note:** Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

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

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

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

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

See Also

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Defining a Password Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.2 Defining a Text Area Answer Control in a Panel

Use this procedure to define a text area answer 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 field unless validation is associated with this answer node.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

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

   Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

Note: The answer name must be unique.

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

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

See Also

Defining a Text Field Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Defining a Password Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.3 Defining a Radio Button Group Answer 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a panel.

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

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

   Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

   Note: The answer name must be unique.

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

   The Edit Data Dictionary dialog box appears.
10. Click the Lookups tab.

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

    The Lookup Entry dialog box appears.
11. In the Display Value field, type the text that appears 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 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 dialog box.
16. Click **OK** to exit the Answer Entry dialog box.
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 dialog box.

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Defining a Password Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.4 Defining a Check Box Group Answer 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a panel.

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

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

   Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.
6. In the Name field, type the name of the answer.

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

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

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Defining a Password Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.5 Defining a Button Answer Control in a Panel

Use this procedure to define a buttons in a script panel.

Caution: The button UI type answer control must only be used in panels that contain *a single node or answer definition.* 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 answer definition is required in this panel.

Steps

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

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

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

Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

Note: The answer name must be unique.

7. From the UI Type list, choose **Button**.
8. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog box appears.
9. In the Lookups tab, select **Specify lookups** and then click **Add**.

   The Lookup Entry dialog box 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 dialog box.
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 dialog box.

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Drop Down List Answer Control in a Panel

Defining a Password Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.6 Defining a Drop Down List Answer 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a panel.

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

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

   Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

   Note: The answer name must be unique.

7. From the UI Type list, choose **Drop Down**.
8. Optionally, in the UI Label field, type the label that appears next to the drop down list in the script panel at runtime.
9. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog box appears.
10. In the Lookups tab, select **Specify lookups** and then click **Add**.

    The Lookup Entry dialog box appears.
11. In the Display Value field, type the text that appears 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 dialog box.
16. Click **OK** to exit the Answer Entry Dialog dialog box.

    The answer appears in the Answer pane in the Properties dialog box.
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 an Answer Definition.
19. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Password Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.7 Defining a Password Answer Control in a Panel

Use this procedure to define a password answer 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 answer control using the Validation command in the data dictionary. This feature is not automatically included in this answer UI type.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

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

   > **Note:** Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

Note: The answer name must be unique.

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

   The answer appears in the Answer pane in the Properties dialog box.
10. If you want to define another 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 an Answer Definition.
12. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.8 Defining a Checkbox Group Answer Control in a Panel

Use this procedure to define a checkbox group answer 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 answer control using the Validation command in the data dictionary. This feature is not automatically included in this answer UI type.

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

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

   Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.

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

   Note: The answer name must be unique.

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

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

10. If you want to define another 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 an Answer Definition.
12. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.14.9 Defining a Multi-Select List Box Answer Control in a Panel

Use this procedure to define a multi-select list box answer control in a script panel. A multi-select list box answer control is the same as the drop down list answer 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 answer control, the multi-select list box answer 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a panel.

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

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

   Note: Every panel is required to have at least one answer defined, and at least one answer definition (or "node") per panel *must* be designated as the default answer for distinct branching. This is true regardless of whether distinct branching is used by the panel.
6. In the Name field, type the name of the answer.

   Note: The answer name must be unique.
7. From the UI Type list, select **Password**.
8. In the UI Label field, type the label that appears next to the text field in the script panel at runtime.

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

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

10. If you want to define another 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 an Answer Definition.

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

See Also

Defining a Text Field Answer Control in a Panel

Defining a Text Area Answer Control in a Panel

Defining a Radio Button Group Answer Control in a Panel

Defining a Check Box Group Answer Control in a Script Panel

Defining a Button Answer Control in a Panel

Defining a Drop Down List Answer Control in a Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15 Working With Answers

After defining an answer for a panel, you can perform the following procedures:

Defining Vertical or Horizontal Answer Control Layout

Adding Validation to an Answer Definition

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Defining Actions

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.1 Defining Vertical or Horizontal Answer Control Layout

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

Prerequisites

Insert a panel onto the canvas.

Steps

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

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

See Also

Adding Validation to an Answer Definition

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

Packaging Java Bean Code Into a JAR File

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.2 Adding Validation to an Answer Definition

Use this procedure to define a validation command to be applied at runtime against an answer provided into the runtime answer control GUI. When the end user provides an answer for this node in a panel, the answer provided will be evaluated based on the parameters dictated by the validation code. In this way, 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.

Prerequisites

- Write the code for the validation command.
- Insert a panel onto the canvas.
- Define an answer control for the script panel.

Steps

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

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. From the Panel Properties dialog, select the answer definition and click **Edit**.

   The Answer Entry dialog box 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 dialog box appears.
7. Define a command for the validation action. (See Defining Commands.)
8. Click **OK** to exit the Data Dictionary dialog box.
9. Click **OK** to exit the Answer Entry Dialog dialog box.
10. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Defining Vertical or Horizontal Answer Control Layout

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.3 Defining Data Constraints for an Answer Control

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

Prerequisites

- Identify the data constraints applicable to this answer
- Insert a panel onto the canvas.
- Define an answer control for the script panel.
-

Steps

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

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. From the Panel Properties dialog, select the answer definition and click **Edit**.

   The Answer Entry dialog box appears.
5. Click **Edit Data Dictionary**.

   The Edit Data Dictionary dialog appears.

6. In the General tab, define the data constraints for the answer control.
7. Click **OK** to exit the Data Dictionary dialog box.
8. Click **OK** to exit the Answer Entry dialog box.
9. Click **Apply** to apply your work or click **OK** to save your work and exit the Properties dialog box.

See Also

Defining Vertical or Horizontal Answer Control Layout

Adding Validation to an Answer Definition

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.4 Reordering Answer Options

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

Prerequisites

- Insert a panel onto the canvas.
- Define an answer control for the script panel.

Steps

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

The Properties dialog box appears.

3. In the Panel tree, select **Answers**.
4. From the Panel Properties dialog, select the answer definition and click **Edit**.

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

See Also

Defining Vertical or Horizontal Answer Control Layout

Adding Validation to an Answer Definition

Defining Data Constraints for an Answer Control

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.5 Reordering Answer Controls

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

Prerequisites

- Insert a panel onto the canvas.
- Define an answer control for the script panel.

Steps

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

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

See Also

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.6 Deleting Answer Options from an Answer Control

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

Prerequisites

- Insert a panel onto the canvas.
- Define an answer control for the script panel.

Steps

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

   The Properties dialog box appears.
3. In the Panel tree, select **Answers**.
4. From the Panel Properties dialog, select the answer definition and click **Edit**.

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

See Also

Defining Vertical or Horizontal Answer Control Layout

Adding Validation to an Answer Definition

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Controls from a Script Panel

Substituting a Java Bean for an Answer

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.7 Deleting Answer Controls from a Script Panel

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

Prerequisites

- Insert a panel onto the canvas.
- Define an answer control for the script panel.

Steps

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

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

See Also

Defining Vertical or Horizontal Answer Control Layout

Adding Validation to an Answer Definition

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Substituting a Java Bean for an Answer

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.15.8 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 the Script Author.

See Also

Defining a Button in a Script Panel

Defining a Text Field in a Script Panel

Defining a Multi Line Text Area in a Script Panel

Defining a Drop Down List Answer Control in a Script Panel

Defining a Radio Button Group Answer Control in a Script

Defining a Check Box Group Answer Control in a Script Panel

Defining Vertical or Horizontal Answer Control Layout

Defining Data Constraints for an Answer Control

Reordering Answer Options

Reordering Answer Controls

Deleting Answer Options from an Answer Control

Deleting Answer Controls from a Script Panel

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.16 Defining Actions

You can perform the following tasks:

Defining Script Pre- and Post-Actions

Defining Panel Pre- and Post-Actions

Defining Group Pre- and Post-Actions

Defining Block Pre- and Post-Actions

Defining Branch Actions

See Also

Defining Panel Answer Controls

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Commands

Defining Shortcuts

Deploying the Script

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.16.1 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 dialog box appears.

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

   The Command dialog box appears.

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

See Also

Defining Panel Pre- and Post-Actions

Defining Group Pre- and Post-Actions

Defining Block Pre- and Post-Actions

Defining Branch Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.16.2 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a panel.

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

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

See Also

Defining Panels

Inserting a Panel

Defining Script Pre- and Post-Actions

Defining Group Pre- and Post-Actions

Defining Block Pre- and Post-Actions

Defining Branch Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.16.3 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a group.

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

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

See Also

Defining Groups

Defining Script Pre- and Post-Actions

Defining Panel Pre- and Post-Actions

Defining Block Pre- and Post-Actions

Defining Branch Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

3.1.16.4 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 tool palette, click the **Toggle Select Mode** tool.
2. On the canvas, double-click a block.

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

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

See Also

Defining Blocks

Defining Script Pre- and Post-Actions

Defining Panel Pre- and Post-Actions

Defining Group Pre- and Post-Actions

Defining Branch Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.16.5 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 tool palette, click the **Toggle Select Mode** tool.

2. On the canvas, double-click a branch.

   The Properties dialog box appears.

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

   The Command dialog box appears.

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

See Also

Defining Branches

Defining Script Pre- and Post-Actions

Defining Panel Pre- and Post-Actions

Defining Group Pre- and Post-Actions

Defining Block Pre- and Post-Actions

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

3.1.17 Defining Commands

You can perform the following tasks:

Invoking a Public Method in a Java Class

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

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

Setting a Constant

Retrieving a Panel Answer From the Scripting Blackboard

See Also

Defining Panel Answer Controls

3.1.17.1 Invoking a Public Method in a Java Class

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

Prerequisites

Write the code that handles the data exchange.

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **Java Command**.
3. In the Command Info area, do the following:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the command string as follows:

      ```
      <fully qualified class name>::<method name>
      ```

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

      The Parameters dialog box appears.

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

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

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

See Also

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

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

Setting a Constant

Retrieving a Panel Answer From the Scripting Blackboard

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.17.2 Invoking a PL/SQL Stored Procedure or Function in a Database

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

Prerequisites

Write the code that handles the data exchange.

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **PL/SQL Command**.
3. In the Command Info area, do the following:
   a. In the Name field, type the name of the command.
   b. In the Command field, type the name of a PL/SQL stored procedure or function in an Oracle database.
4. If you want to list parameters for the command object, then, in the Parameters area, do the following:
   a. Click **Add**.

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

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

   e. In the In/Out list, select the parameter type.
   f. Click **OK** to exit the Parameters dialog box.
   g. If you want to add another parameter, then repeat steps a through f.

5. If you want to connect to the Oracle database using new connection parameters, then do the following:
   a. Clear the **Reuse an existing connection** box.
   b. Type the connection parameters. (See Database Connection Guidelines.)
6. If you want to reuse existing connection parameters to connect to the Oracle database schema, then do the following:
   a. Select the **Reuse an existing connection** box.
   b. From the Existing Connections list, select a connection.
   c. In the Password field, type the password for the selected database connection.
7. Click **OK** to exit the Command dialog box.
8. Save your work.

See Also

Invoking a Public Method in a Java Class

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

Setting a Constant

Retrieving a Panel Answer From the Scripting Blackboard

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.17.3 Invoking a PL/SQL Procedure or Function in a Forms Server

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

Prerequisites

Write the code that handles the data exchange.

Steps

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

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

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

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

      The Parameters dialog box appears.

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

See Also

Invoking a Public Method in a Java Class

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

Setting a Constant

Retrieving a Panel Answer From the Scripting Blackboard

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.17.4 Setting a Constant

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

Prerequisites

None

Steps

1. Navigate to a Command dialog box.
2. In the Command Type area, select **Constant Command**.
3. In the Command Info area, type the name of the command in the Name field.
4. In the Return Value area, do the following:
   a. Click **Edit**.

      The Parameters dialog box appears.
   b. In the Value field, type the return value.

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

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

See Also

Invoking a Public Method in a Java Class

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

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

Retrieving a Panel Answer From the Scripting Blackboard

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.17.5 Retrieving a Panel Answer From the Scripting Blackboard

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

Prerequisites

None

Steps

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

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

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

See Also

Invoking a Public Method in a Java Class

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

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

Setting a Constant

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.18 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. If not already logged into the database, accessing the command library requires you to enter database connection parameters for the specific database instance of the Oracle Scripting command library you wish to use. You will remain connected for the duration of the Script Author session.

Note: To make a command available for reuse in any script, you must first enter your command in the Script Author command library to make it available to all scripts. If you define a command in a specific script instead of the command library, there is currently no method to copy or move that defined command to the command library.

You can perform the following tasks:

3.1.18.1 Defining a Reusable Command

Defining a command in the command library makes it available for reuse to script developers using the Script Author in the same database instance. Use this procedure to define a Script Author command in the command library.

You can create a command in the command library from the Script Author at any time, and import it to the appropriate location in the script. Select Command Library from the Tools menu. Enter database connection information for the database instance of the Oracle Scripting command library you wish to access (required once per Script Author session). In the Command Library dialog, click Add, and define your command as usual. This places the command in the command library for immediate or subsequent reuse.

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

Prerequisites

None

Steps

1. From the Tools menu, select Command Library.

   If you are already logged into the database, the Command Library window appears.

   If you are not already logged into the database, the Database Connection Info dialog appears. A database connection is required in order to store the command in the database.

2. If required, enter the appropriate connection information to connect to the database instance in which you want to store the reusable command.

   For more information, see Connecting to the Database.

   The Command Library window appears.

3. To begin adding a command to the command library, click **Add**.

   The Command dialog box appears.

4. Enter all command parameters.
5. Click **OK** in the Command dialog box to save your work.

   The command is now stored in the library as a reusable command.
6. When you have completed your tasks in the command library, click**Close** to exit the command library window.
7. Save your work.

See Also

Section 3.1.18.1, "Defining a Reusable Command"

Section 3.1.18.2, "Copying, Modifying, or Deleting a Reusable Command"

Section 3.1.18.3, "Importing a Reusable Command Into a Script"

3.1.18.2 Copying, Modifying, or Deleting a Reusable Command

Prerequisites

A reusable Script Author command must be stored in the applications database.

Steps

1. From the Tools menu, select Command Library.

   If you are already logged into the database, the Command Library window appears.

   If you are not already logged into the database, the Database Connection Info dialog appears. A database connection is required in order to store the command in the database.
2. If required, enter the appropriate connection information to connect to the database instance in which you want to store the reusable command.

   For more information, see Connecting to the Database.

   The Command Library window appears.
3. 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".

4. 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 dialog box 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.

5. 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.

6. When you have completed your tasks in the command library, click**Close** to exit the command library window.

7. Save your work.

3.1.18.3 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. Navigate to a Command dialog box.
2. Click **Use Library Command**.

   If you are already logged into the database, the Choose Command From Library dialog appears.

   If you are not already logged into the database, the Database Connection Info dialog appears. A database connection is required in order to store the command in the database.
3. If required, enter the appropriate connection information to connect to the database instance from which you want to retrieve the reusable command.

   For more information, see Connecting to the Database.

   The Choose Command From Library dialog appears.
4. 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 dialog box 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.
5. 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.

6. Click **OK** to exit the Command dialog box.
7. Save your work.

3.1.19 Deploying the Script

You can perform the following tasks:

Checking Script Syntax

Deploying a Script to the Database

Connecting to the Database

See Also

Getting Started with the Script Author

Obtaining Script Requirements Before Creating a Script

Defining Global Script Attributes

Working with Script Files

Working with the Tool Palette

Viewing Objects on the Canvas

Working With Objects on the Canvas

Defining Panels

Defining Groups

Defining Blocks

Defining Branches

Controlling Script Flow

Defining Panel Answer Controls

Working With Answers

Defining Actions

Defining Commands

Defining Shortcuts

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.19.1 Checking Script Syntax

Use this procedure to check the syntax of a script.

Prerequisites

Create or open a script.

Steps

1. Choose **Tools > Syntax Check** or click the **Check Syntax** icon in the toolbar.

   The **Check Syntax** icon is the fourth icon from the right of the toolbar, located immediately below the menus in the Script Author interface.

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

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

See Also

Deploying a Script to the Database

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.19.2 Deploying a Script to the Database

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

Prerequisites

- You must have a syntactically correct script.
- You will need the database hostname, TNS port and SID.
- Open a script to be deployed in the 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 is the third icon from the right of the toolbar, located immediately below the menus in the Script Author interface.

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

   Note: Always use a new connection.

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

See Also

Defining the Script Name

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.1.19.3 Connecting to the Database

When deploying a script, performing a query, insert, or update command, or performing a PL/SQL command, you will need to use the Script Author interface to connect to the database.

Enter database connection information for your environment based on the following steps and guidelines.

Steps

1. Optionally, in the Connection Name field, enter a name for this connection.

   This information is used by the Script Author only. Any characters may be used as part of this name. Defining a connection name is optional.

2. In the Host Name field, enter the name of the database host (including domain).

   For example, type **database.vision.com**.

3. In the Port Number field, enter the TNS port of the database.
4. In the SID field, enter the database global name or SID.
5. In the User Name field, enter an appropriate database User Name.

   This user must have privileges to access the IES_DEPLOYED_SCRIPTS table of the applications database (for example, the apps user).

6. In the Password field, enter the password for the appropriate user name.
7. Once you have entered all appropriate connection information, click **OK**.

Follow these database connection guidelines.

Database Connection Guidelines

Database connections are made from Scripting using thin JDBC only. When you invoke a connect object, the following JDBC driver is invoked automatically, specifying the Java class to be used as the driver for connecting to the Oracle database:

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

Connection Name. The connection name is used by Script Author only and can contain any characters. Defining a connection name is optional.

Host Name. The host name parameter specifies the name of the database server machine or host. Including the domain when specifying the host name is recommended.

Port Number. The port number parameter specifies the port number for TNS connections. This value is determined by the configuration of the database.

SID. The SID parameter specifies the SID of the database or the instance name.

User Name Use the Oracle8*i* or Oracle9*i* database administrator login.

Password Use the apps level administrator password.

See Also

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.2 Troubleshooting the Script Author

This topic includes the following topic groups:

Obtaining Help in the Script Author

Storing Strings in the Script Author

Known Issues Using COMMIT in Custom PL/SQL Procedures

See Also

Using the Script Author

Using the Scripting Engine

Taking a Survey

Using the Survey Admininstration Console

3.2.1 Obtaining Help in the Script Author

Script Author 11*i* includes online help. However, at the current time, this online help is not accessible from the Help menu of the Script Author. To view online help, perform the steps indicated below.

Prerequisites

- Online help has been updated as of release 11.5.6. You must access online help from a computer loaded with the Script Author tool.
- You must use a Web browser to access online help.

Steps

1. On your local computer or network, navigate to the directory in which the Script Author was installed. As an example, the path to access this directory on your computer may be similar to the following:

   ```
   C:\OraHome1\oracle\apps\ies\author
   ```

   Note: *The path to the ScriptAuthor directory may be different on your computer.* The Script Author may be located in your Oracle Home directory, although this is not a requirement.

   To locate your Oracle Home directory on Windows NT: (1) Right click on My Computer from your desktop; (2) Click the Environment tab; (3) In the Variable column, scroll down to view the Path variable; (4) Note the Oracle Home directory listed in the Path variable.

   To locate your Oracle Home directory on Windows 2000: (1) Right-click on My Computer from your desktop, and select Properties; (2) From the System Properties dialog, click the Advanced tab; (3) In the Environment Variables area, click Environment Variables; (4) In the Variable column, scroll down to view the Path variable; (5) Note the Oracle Home directory listed in the Path variable.

2. Navigate to the **docs** subdirectory.
3. From your Web browser, open the Script Author Online Help table of contents, TOC.HTM, from the **docs** subdirectory.

4. In your Web browser, navigate to the appropriate topic as applicable.

See Also

Storing Strings in the Script Author

Known Issues Using COMMIT in Custom PL/SQL Procedures

3.2.2 Storing Strings in the Script Author

A file called BUNDLE.CONTENTS is used to store strings in the Script Author. When the Script Author is launched, it checks to see if the file exists--if it does, then the Script Author pulls its strings from there.

If the file doesn't exist, the Script Author will prompt you to either load strings from the database or install default. Default is a set of english strings (EN.CONTENTS) that ship with the Script Author for running the Scripting Engine in stand-alone mode for testing, etc.

See Also

Obtaining Help in the Script Author

Known Issues Using COMMIT in Custom PL/SQL Procedures

3.2.3 Known Issues Using COMMIT in Custom PL/SQL Procedures

Oracle Scripting manages the transaction scope throughout each script interaction. When the script first starts, the transaction opens, and when the script runs to completion, Scripting automatically commits and closes the transaction. This is true regardless of whether Scripting is executing in the Scripting Engine or as a survey in a Web browser.

In between, any code called on the Scripting connection may not call the PL/SQL COMMIT statement. If a COMMIT is called by a custom PL/SQL procedure, problems may occur if an agent in an interaction center or a respondent taking a survey goes back and changes an answer. Basically, as SQL is issued to the database, Scripting issues savepoints to allow for rollback in case the script end user goes back in the flow of the script to change an answer. If the new answer causes the flow of the script to take a different branch, a rollback to the appropriate savepoint is called. However, if a COMMIT call was made, then the database will throw an error on the attempt to roll back.

The best way to prevent the problem is not to make any COMMIT calls anywhere during a scripting interaction. However, in some cases, this is not possible. There are two possible changes that can be done to handle this issue.

1. The first is the "Autonomous transaction." This allows an independent, nested transaction to be started within the main transaction. Thus, when the COMMIT call is made, only those calls in the autonomous transaction are saved. When the agent changes answers in the rest of the script, even before the autonomous transaction, and a rollback is issued, the database will not give an error. However, the SQL calls that were made within the autonomous transaction *will not be rolled back.*
2. In addition, Oracle Scripting provides the ability to script user from changing answers in earlier panels. The two API calls, deactivateAllPrevious Panels() and deactivateAllPreviousPanels(StringdeactivationMessage), will prevent the agent from re-enabling any panel from the first panel to the current panel (when the API is called). In addition, specifying a deactivationMessage parameter will cause the Scripting Engine to display the deactivationMessage when the agent attempts to re-enable any panel that has been deactivated.

3.3 Using the Scripting Engine

Scripts created in the 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. Oracle Scripting 11*i* is integrated with three business applications in release 11.5.8: the Customer Support module of Oracle TeleService, Oracle TeleSales, and Oracle Collections. For script testing purposes, scripts can also be launched in stand-alone mode using the Script User or Scripting Agent responsibilities. For information on these responsibilities, see Responsibilities That Can Launch the Scripting Engine.

The Scripting Engine GUI is simple to use. An agent can be thoroughly trained in a very short time. The GUI contains the following features:

- The Panel Display Area
- The Progress Area
- The Script Information Area
- The Shortcut Button Bar
- The Disconnect Button
- The Toolbar

The GUI elements listed above are described in detail in the section Understanding the Scripting Engine. As a summary, when the Scripting agent launches a script, a separate Forms window opens with the Scripting Engine GUI. The agent navigates from panel to panel in the Panel Display Area. Script Author objects such as groups, blocks, Start and Termination nodes and the various branch types provide functionality behind the scenes; they do not display.

Each panel requires agent interaction (typically by clicking the answer controls and clicking the panel's **Continue** button). Each answer control that requires a click also has one or more keyboard command equivalents. For example, pressing the space key for any answer control confirms that selection (as if the user had clicked the mouse). Pressing the tab key cycles through the individual lookup choices for a radio button. Pressing the tab key also cycles through the individual lookup choices for a checkbox or a checkbox group, and the space key toggles the selection on and off. Pressing the tab key selects a dropdown list control. The Up or Down arrow changes the dropdown choice selected, and the space key confirms the dropdown selection. Pressing the Ctrl key (Option key for Macintosh OS) while making your selection allows the end user to select more than one checkbox in a checkbox group or more than one dropdown in a multi-select dropdown list.

An agent can click on and visit a previously displayed panel by using the Toolbar controls or by clicking a panel label in the Progress Area.

Optionally, a script may include information above the Panel Display Area in the Script Information Area, which has nine possible field values, each able to have its own label. These values may be static or dynamic, and may contain alphanumeric information or a timer. Below this (if present) and immediately above the Panel Display Area, a set of Shortcut buttons may optionally be programmed into the script. These will jump a user to a section of the script (based on a group's Shortcut property) or fire a Scripting API (for example, to start or stop a timer in the Script Information Area). If enabled, the Disconnect button will "jump" the user to whatever group contains the shortcut property "WrapUpShortcut."

See Also

Using the Script Author

Troubleshooting the Script Author

Taking a Survey

Using the Survey Admininstration Console

Understanding the Scripting Engine

Scripting Engine Users

Responsibilities That Can Launch the Scripting Engine

3.4 Taking a Survey

There are a variety of ways to obtain survey respondents. (These methods are discussed in the section Survey Respondent Entry Points.) For list-based survey campaigns, the typical method is to send an e-mail invitation with the Survey deployment-specific URL (typically as a hyperlink). List-based survey campaign deployment URLs also contain unique respondent IDs which can control the amount of times a respondent may take a survey, and enables tracking of respondents against all list members invited to participate.

Non-list-based survey campaigns include a URL (which, depending on the method of delivery, may be a hyperlink) that a user may choose to visit. Users that choose not only to visit the ULR but also to participate in the survey are survey respondents. There is no respondent ID for non-list-based respondents, although iSupport respondents may be tracked using unique iSupport identification codes or cookies.

Regardless of the entry point, the respondent must use an Oracle Applications 11*i*-compliant Web browser to take the survey. When they navigate to the appropriate URL, they will see the header JSP resource at the top of each page, followed by any panel text and answer controls. Each HTML page that displays equates to a single panel in the Script Author. Just as in the Scripting Engine model, Script Author objects such as groups, blocks, Start and Termination nodes and the various branch types provide functionality behind the scenes; they do not display.

Each HTML page requires agent interaction (typically by clicking the answer controls and selecting the panel's **Continue** button). If a Java stack error results (typically due to environmental factors at the enterprise), the error JSP resource page will display. Upon completion of the last panel in the script, the final JSP resource page will display.

Upon viewing the final JSP survey resource, the survey has been completed and the information is immediately committed to the Scripting schema in the Oracle Applications database. Concurrent processing then makes summarized survey information from the Scripting Schema available to the **Analysis** tab of the Survey Admininstration console for subsequent reporting and analysis. This is transparent to the user. Upon reaching the final JSP survey resource page, the respondent may exit the browser, surf to any links contained on that HTML page, and so forth. Their information has been successfully captured. For list-based survey campaigns, their response will immediately be available to view from the **Response** tab of the Survey Admininstration console.

See Also

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Using the Survey Admininstration Console

3.5 Using the Survey Admininstration Console

Using the Survey Admininstration console, a survey administrator can define survey campaign information, create cycles and deployments, define JSP resources, deploy surveys, monitor individual responses to an active or formerly active deployment, and generate reports based on summarized respondent information.

The Survey Admininstration console is a JSP/HTML-based survey campaign administrative GUI. Users with the Survey Administrator Oracle Applications responsibility can access this console from the Oracle eBusiness Center HTML login.

Substantial step-by-step procedures for accomplishing particular administration tasks are discussed in the section Administering Oracle Scripting Survey Campaigns in *Oracle Scripting Implementation Guide, release 11i*. This section discusses applications and theory regarding the use of the Survey Admininstration console.

Topics in this topic group include:

- Campaigns and the Survey Component of Oracle Scripting
- Survey Campaign, Cycle, Deployment Hierarchy
- Survey Media Channels
- Survey Respondent Entry Points

See Also

Using the Script Author

Troubleshooting the Script Author

Using the Scripting Engine

Taking a Survey

3.5.1 Campaigns and the Survey Component of Oracle Scripting

A **campaign** can be defined as a focused effort to achieve a particular goal from a targeted population over a specific period of time for a particular business purpose. A campaign is a concept often employed in interaction centers.

From a Survey component perspective, the goal of a campaign is to obtain measurable, actionable information. This information allows the organization to analyze its customer/prospect responses to key initiatives, existing level of service, and so forth. The vehicle for obtaining this information is a survey questionnaire (created in the Script Author, as discussed in the next section). There is a one-to-one correspondence between a survey campaign and a specific script.

The Survey component employs an administrative console using Java Server Pages technology built on the JTF HTML-based technology stack. A survey administrator can log into the Survey Admininstration console (from the eBusiness Center login) to define and configure survey campaign information. Prior to defining the survey campaign:

- The survey campaign information must be defined by a campaign administrator and other relevant individuals within the interaction center or enterprise;
- The Survey component of Oracle Scripting (and related components) must be appropriately implemented;
- A customized **survey questionnaire** must be developed and deployed.

See Also

- Survey Campaign, Cycle, Deployment Hierarchy
- Survey Media Channels
- Survey Respondent Entry Points

3.5.2 Survey Campaign, Cycle, Deployment Hierarchy

The hierarchy for defining a survey campaign in the Survey Admininstration console is:

- Survey Campaign
- Cycle
- Deployment

Survey Campaign

The survey campaign, at the top level of the hierarchy, is where parameters affecting the entire survey campaign are administered, such as the specific script to be used as the survey questionnaire, the survey campaign name, a description of the survey, status (open and cancelled), and survey resources. A survey campaign contains one or more **cycles** (described below).

Cycle

Cycle properties include a cycle name, an anonymous flag, a minimum response percentage, and status (open, active, or cancelled). Each cycle contains one or more **deployments** (described below). 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.

Deployment

A deployment is the lowest member of the survey hierarchy. It is also the most granular. Deployments belong to a particular cycle. Deployment properties include list (if any), media type, the parent survey campaign and cycle, status (incomplete, active, closed), owner (Survey administrator login), deploy date, response end date, and last update date. 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 participate in a survey campaign.

List-Based Information

The **Create Deployment** page of the Survey Admininstration console also contains fields in which survey administrators identify information specific to list-based survey campaigns. If you fill out list information on this page, the deployment will be considered list-based; otherwise it will be considered a non-list-based deployment.

Caution: For current releases, if you designate a deployment as list-based, this property has a backwards-cascading effect, causing the entire survey campaign to be a list-based campaign.

Future releases are expected to allow both list-based and non-list-based deployments in a single survey campaign.

The single campaign-level property configured here is the Web Server Uniform Resource Locator (URL). This will auto-populate with the URL of the current Apache Web server session (used to log into the eBusiness Center to access the Survey Admininstration console). If this value is incorrect or fails to auto-populate, you must enter the appropriate Apache Web server, port, and OA_HTML bin. If using a secondary Apache Web server, you must define this secondary Web server here.

Note: When entering data into the **Web Server URL** field, include the name of the Web server, port, and reference to HTML bin (ending with a slash). For example: http://www.company.com:8888/OA_HTML/

The remaining properties are specific to the deployment and include the list name, invitation and reminder template names, the maximum number of responses to the survey allowed per person, the minimum number of responses enforced prior to closing the deployment, and the subject line of the e-mail invitation. For deployments using reminders, the number of reminders and reminder interval (in days) is configured here. Otherwise, leave this information blank.

See Also

- Administering Oracle Scripting Survey Campaigns > Defining Deployments in *Oracle Scripting Implementation Guide, release 11i*
- Campaigns and the Survey Component of Oracle Scripting
- Survey Media Channels
- Survey Respondent Entry Points

3.5.3 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

- Campaigns and the Survey Component of Oracle Scripting
- Survey Campaign, Cycle, Deployment Hierarchy
- Survey Respondent Entry Points

3.5.4 Survey Respondent Entry Points

Surveys range in goals from measuring customer satisfaction to obtaining market research, providing benchmark data, measuring political or public opinion, and so on. Populations targeted for participation in a survey vary according to the business drivers and business purpose for obtaining the data.

Sometimes the survey **sample** (the population from which information is solicited) is clearly defined, either from an existing list or an identifiable transaction session, and other times it is more random. In current releases of the Survey component, there are four supportable entry points for survey respondents, falling into two categories: list-based and non-list-based.

3.5.4.1 List-Based Entry Points

Some survey campaigns require an already-identified target population. Business rules underlying these survey campaigns may require respondents to be customers or prospects who have already been served by or otherwise have had initial contact with the enterprise. Oracle Marketing Online is a tool to manage lists of customers or prospects which can be leveraged for survey campaigns.

The single list-based entry point for survey respondents is a scenario in which Fulfillment and OMO are implemented at the enterprise. Lists created in OMO are employed using Fulfillment APIs to send **invitations** (e-mail messages inviting list members to take part in a survey). Those list members who choose to participate in the survey (**respondents**) click on a URL or hyperlinked invitation image in the e-mail message, and are directed to the appropriate survey questionnaire in a Web browser.

List-based entry points offer survey campaign administrators the ability, at the cycle level, to monitor and track survey responses down to an individual respondent if desired, or to designate the campaign as anonymous.

List-based survey campaigns can continue to leverage Fulfillment and OMO integration using **reminders**. With reminders, follow-up e-mail messages can be sent to list members reminding them of the deadline for participating in the survey.

If a particular cycle was designated as not anonymous (i.e., the Anonymous flag was set to No), reminders will be sent only to those list members who have not yet responded. If the cycle was designated as anonymous, all list members (respondents and the remaining pool of potential respondents) will receive reminder e-mail messages. At this time, reminders are the only aspect affected by the Anonymous flag.

3.5.4.2 Non-List-Based Entry Points

The three non-list-based entry points for iSurvey respondents include Oracle iSupport, "pop-up" surveys from an enterprise Web site, and "walk-through" Web browsing from non-targeted members of the general population.

Oracle iSupport

Oracle iSupport is a self-service web application that enables enterprises to provide a method for their customers to log service or maintenance requests. If a Survey hyperlink is added to the iSupport interface, those utilizing iSupport can provide immediate feedback via iSurvey regarding their iSupport customer experience. As an added benefit, iSurvey can capture and retain the iSupport user ID assigned when the customer logged in, and tie the responses to that iSupport user, enabling some level of tracking between the two applications.

Pop-Up Surveys

Some campaigns may be designed to solicit feedback from individuals visiting an enterprise’s Web site by adding "pop-up" surveys, using JavaScript or other technology to pop a separate window inviting the Web site visitor to take a short survey. The visitor has the option to accept or decline the invitation. The choice to accept is the hyperlink to launch the survey, whereas the choice to decline closes the pop-up window and allows the visitor to continue viewing the Web site.

Walk-Through Surveys

The Survey component of Oracle Scripting provides the added ability to obtain survey data from a wide audience. Some campaigns are best served by soliciting feedback from a random sample of the general population. This would best be served by "walk-through" traffic, with a sample population of respondents gathered from banner advertisements on Web sites and by respondents directed to the iSurvey URL from print and broadcast media. For enterprises with substantial Web traffic, walk-through survey data can also be obtained by simply providing a survey link on the enterprise Web site.

As opposed to list-based respondents who are responding to an invitation, or iSupport customers, there is no standard tracking mechanism for respondents whose entry point is a Web site either from a pop-up survey or as a walk-in survey respondent. Nor are lists, invitations or reminders relevant, precluding the need for OMO and Fulfillment as well as reliance upon OES.

See Also

- Understanding the Survey Component > The Survey Admininstration Console
- Campaigns and the Survey Component of Oracle Scripting
- Survey Campaign, Cycle, Deployment Hierarchy
- Survey Media Channels