Oracle® Scripting

Implementation Guide

Release 11*i*

April 2001

Part No. A86115-04

ORACLE®

Oracle Scripting Implementation Guide, Release 11*i*

Part No. A86115-04

Contents

Using Oracle Scripting

Integrating Oracle Scripting

Related Products and Components

Planning an Oracle Scripting Implementation

Implementing Oracle Scripting

Testing Oracle Scripting

Troubleshooting Oracle Scripting

C Scripting Project Scoping and Discovery

D Modifying `appsweb.cfg` File for Scripting Implementations

Send Us Your Comments

Oracle Scripting Implementation Guide, Release 11*i*

Part No. A86115-04

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

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

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

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

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

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

Preface

Welcome to Oracle Scripting, Release 11*i*, part of the Oracle Customer Relationship Management (CRM) Interaction Center suite of applications.

This Oracle Scripting Implementation Guide provides information and instructions to help you work effectively with Oracle Scripting.

This preface explains how the Oracle Scripting Implementation Guide is organized and introduces other sources of information that can help you.

Intended Audience

This guide is a reference for all who will implement Oracle Scripting:

- Management and staff of an enterprise (typically a call center or interaction center) about to implement Oracle Scripting
- Members of an enterprise that need to modify existing scripts and deploy them in place of or alongside existing scripts in the interaction center
- Oracle Consulting Services (OCS) management responsible for scoping, planning and resourcing the implementation of products in the Oracle technology stack
- OCS technical staff responsible for implementing products in the Oracle technology stack
- Oracle Support Services (OSS) staff responsible for supporting implementations of Oracle Scripting
- Companies partnering with Oracle Corporation to provide enterprises with consulting in areas of management, implementation, or integration of software systems.

This guide assumes you meet the following prerequisites:

1. Understanding of the company business processes, products and services.
2. Requirement to implement Oracle Scripting.
3. Basic understanding of Oracle Scripting, Oracle Applications and Oracle *8i* Relational Database Management System (RDBMS).
4. Access to staff with knowledge in Oracle 8*i* RDMBS administration (including administration of Java in the database), background in SQL, PL/SQL, SQL* Plus programming, and field engineering (software installation, implementation and configuration).

Structure

This manual contains the following chapters:

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

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

“Integrating Oracle Scripting” provides information on Oracle Scripting’s degree of integration with other Oracle applications, and briefly discusses different application integration approaches, including screenshots and samples.

“Related Products and Components“ provides a discussion of other software products or components that affect the implementation and ongoing operation of Oracle Scripting.

“Planning an Oracle Scripting Implementation” presents the concepts and considerations necessary to help determine your optimal configuration and implementation strategy.

“Implementing Oracle Scripting” provides general descriptions of the setup and configuration tasks required to implement the application successfully.

“Testing Oracle Scripting” provides discussion on how to test an Oracle Scripting implementation from the standpoint of manual post-installation implementation steps, custom script development testing, and integration testing.

“Troubleshooting Oracle Scripting” provides discussion on how to troubleshoot a variety of scripting issues - implementing scripting, launching scripts, etc.

“Related Documentation and Resources” provides detailed references to printed and online resources for those who need to implement CRM 11*i* Oracle Applications in general and Oracle Scripting in particular, as well as additional resources for all users of Scripting. For the additional benefit of Oracle and Oracle Partner consultants and support staff, specific reference is also made to internal-only resources.

This manual also includes the following appendices:

“Acronym Glossary“provides quick-lookup capabilities to decipher the many acronyms included in this volume.

“Oracle Scripting Tutorial Exercises“ contains a set of five exercises to familiarize those who need to implement Scripting with the processes involved in building, deploying and executing simple scripts (with no Java commands).

“Scripting Project Scoping and Discovery“amplifies discussions regarding business rules specific for Oracle Scripting to help provide Oracle consultants and consulting partners with additional tools to scope prospective Scripting engagements and to help them perform adequate Discovery prior to beginning an implementation.

“Modifying `appsweb.cfg` File for Scripting Implementations“ discusses in detail the process of modifying the Web server configuration file that causes Java Archive files to cache on a client workstation.

In order for this edition of the Oracle Scripting Implementation Guide to be equally useful to a wide audience, there are several entry points.

- For those who are planning an implementation and need to consider what factors are involved, it is recommended you skim through the document in its entirety.
- For those who are currently in process of implementing this software, refer first to the implementation starting point section, continue on to the implementation checklist, and bear in mind there is an acronym glossary, troubleshooting section, and detailed list of references.
- For those who are implementing this software and are not following the recommendations as listed in the implementation starting point section, you may need to refer most often to the troubleshooting section and the detailed list of references.
- Project managers should carefully examine the Considerations for Planning an Oracle Scripting Implementation section.
- Anyone converting a 3-tiered JServer Technology Stack implementation to a Caching Architecture will benefit from the new material on these requirements.

- All users stand to benefit from skimming the Implementation Roles and Functions section to clearly understand the different roles, skills, and level of coordination required for a successful implementation.

Note that the Portable Document Format (PDF) version of this document contains hyperlinks that help you navigate easily to different portions of the document.

New for this version is a table of contents for those not viewing the Scripting IG in PDF format.

Related Documents

See the last section prior to appendices, Related Documentation and Resources, for a list of references.

Understanding Oracle Scripting

Oracle eBusiness Suite of Applications and Oracle Scripting

Oracle Scripting is one application module of Oracle's Call Center Suite of Applications, a series of robust solutions designed to meet the critical business needs of today's advanced Customer Relationship Management (CRM) and Interaction Center environments. Together, CRM and Enterprise Resource Planning (ERP) applications comprise Oracle's eBusiness Suite of applications.

Oracle Scripting presents scripted messages to guide call center / interaction center agents through their interactions with customers. In addition to its easy-to-use thin client agent Graphic User Interface (GUI), it contains a powerful authoring environment of graphical layout tools to create, modify, and deploy scripts.

Terminology

Scripting and Agents

Scripting is a generic term for the concept of providing customer service representatives or "agents" a method of guided interaction with a customer or prospect.

Scripts and Call Guides

Regardless of whether a script is typed on paper or presented to an agent through a software application, in a call center environment where agents interact with customers over the telephone it is often referred to as a "call guide." (This document uses both terms interchangeably.)

Call Centers and Interaction Centers

Call centers are increasingly using other methods to satisfy customers (email, the Internet, etc.), and are now often referred to as “customer interaction centers.“ (This document uses both terms interchangeably.)

Scripting Concepts

The primary goal of a script is to provide customer interaction center agents with the tools they need to easily respond to or present information to customers in a consistent manner. Through defining effective scripts, customer interaction centers can reduce the time it takes to train agents, make the agents more effective and productive while they are on the telephone with customers, and ensure that all agents have the same source of information at their fingertips.

Oracle’s Scripting Solution

Scripting applications automate the concept of scripting by incorporating business rules into a software application. Typically, a script consists of a series of presentation panels, containing one or more questions, which guide an agent through a conversation with a customer or prospect. A script may be as simple as a consecutive set of questions to be asked (such as a survey), or it can be a fairly complex set of questions and answers that guide an agent, based on context, to appropriate screens within an application.

Oracle Scripting is a set of script development and execution tools designed to meet the needs of call centers across a broad range of customer information needs.

What Are the Components of Oracle Scripting?

Oracle Scripting consists of two components:

- The **Scripting Engine**, an easy-to-use GUI at the agent desktop which guides an agent through a script. This is the runtime component used by interaction center agents. Based on the script design, an agent’s path through each instantiation of a script (referred to as an *interaction*) may be unique, based on answers that are evaluated at runtime. Alternatively, the flow of the script may be tightly controlled, enforcing uniformity scripting interaction.
- The **Script Author**, a script development tool which provides a visual layout for trained consultants or interaction center staff to lay scripts out graphically, taking business requirements and implementing them in a functional script or call guide.

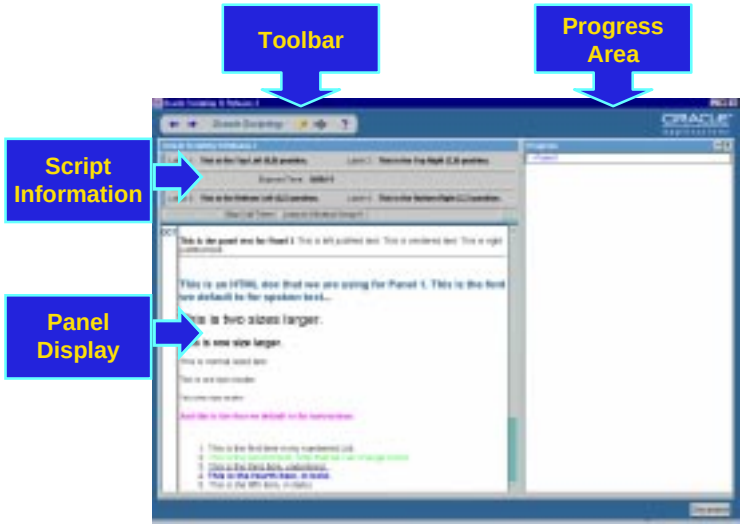


Figure 1. Quick view of the Scripting Engine

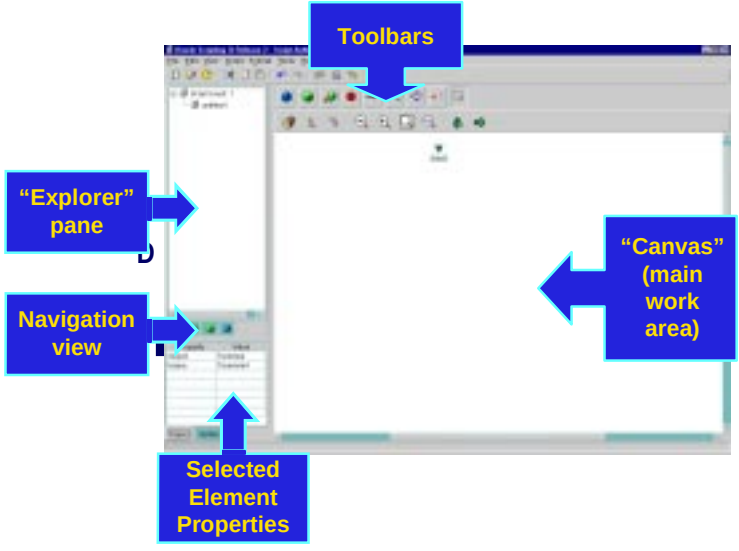


Figure 2. Quick view of the Script Author

Thus, Scripting provides both a development environment for interaction centers, and a runtime environment in which agents are guided through their conversations with customers using an intuitive GUI. As the customer interacts with an agent, the agent records each answer in the Scripting Engine, and progresses to appropriate text screens. Movement through the script may be determined dynamically based on answers provided, or sequentially through all screens in the call guide, as determined by the logic programmed into the script using the Script Author. Thus, as agents record answers they are automatically presented with the next screen, appropriate to the context of the conversation. That screen might contain information solicited by the customer that the agent needs in order to respond fully and accurately, or a question the agent should ask subsequent to previous answers provided to the agent.

Scripts can be created with multiple skill levels in mind, enabling a call center to provide detailed scripts to novice agents, and more flexible routing throughout the script to more experienced agents. (In future releases, pre-defined views and information will be presented based on the agent's responsibility code.) Either method enables a script author to create a single script that satisfies several agent groups. The script can be created in a timely manner and deployed to the Oracle Applications database for use by all agents as appropriate.

Oracle Scripting is a thin-client application supporting multi-tiered architectural models on Oracle 8*i*. It provides a unified desktop for multiple external systems because it can obtain data from those systems and display the data in a logical, consistent GUI. In most call centers, agents are required to deal with up to twenty different applications when interacting with customers. Oracle Scripting can guide the agent's workflow among these systems by pre-positioning the most important customer data where it is easy to find. This can result in significantly reduced training time, increased agent productivity, enhanced service and reduced cost of call center operations.

Where Oracle Scripting Resides Within the Overall CRM 11*i* Footprint

Oracle Scripting is part of the Call Center family of products. It can be considered a "tool" rather than an application, because it provides Authoring tools that allow trained users to customize their scripts for different agents, applications, and campaigns.

In fact, while there are some Best Practices Scripts which represent generic credit card, collections, and telesales interactions and will be accessible "out of the box" with the TeleSales and TeleService (the Customer Support module), it is likely that most Oracle customers will customize the scripts they use for their specific

interaction center requirements. This customization effort will typically be associated with some Consulting engagement as part of the implementation of other CRM products. Note also that, as indicated in the Current Integrations Between Scripting and Other CRM Applications section below, such customized efforts are not supported by Oracle Support Services (OSS) unless contracted specifically.

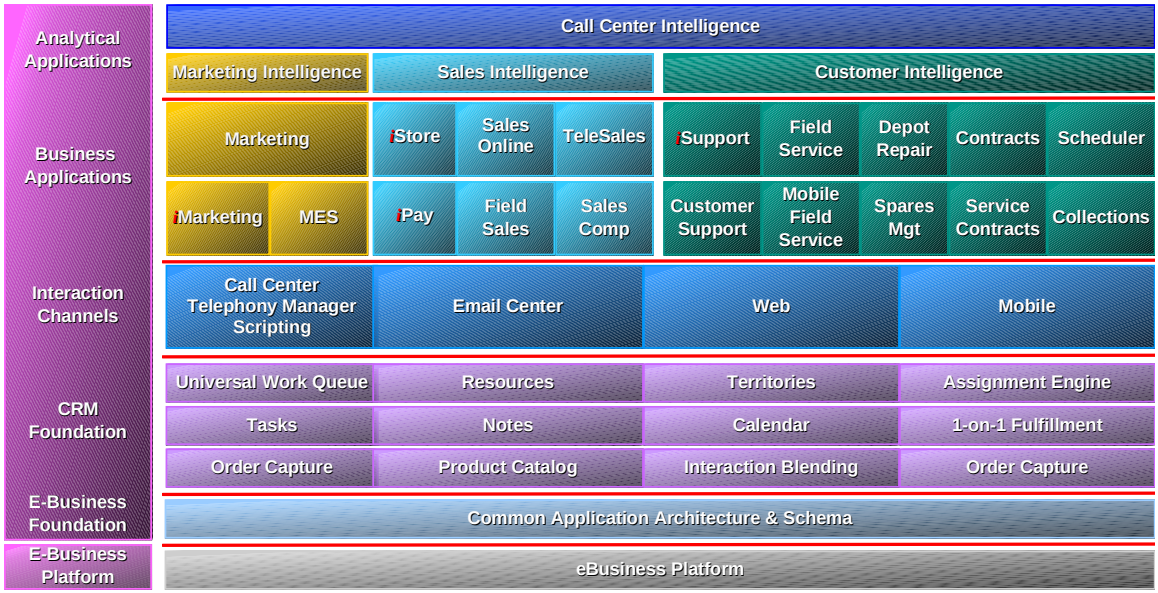


Figure 3. The overall CRM 11i footprint

Supported Platforms

Oracle Scripting Script Author

At the time of this writing, the Script Author is only supported on the Windows platform (Windows NT, 95, 98, 2000). Although Oracle Scripting Script Author 11*i* has been certified on the Apple Macintosh platform, it has not been released, as no standards have been developed for installation on Macintosh OS client workstations at this time. Consequently the Script Author is not generally available on any platform other than Windows.

Oracle Scripting Engine

From a server perspective, Oracle Scripting Engine is supported for both Windows NT and some UNIX platforms. Scripting 11*i* relies on an Oracle 8*i* database. The most common platforms for 8*i* servers are Solaris and NT, although other platforms are also supported.

Note: The latest information on supported operating systems, as well as information on releases, patches, or supporting software products can always be found at Oracle MetaLink at http://metalink.oracle.com.

How Scripting Differs From Other Oracle CRM Applications

Scripting Is Not Intended for Stand-Alone Use

Because Oracle Scripting is targeted at a very specific subset of customer interaction center requirements, enterprises intending to use this application tool set should implement Oracle Scripting as a *part* of another, broader set of CRM applications. Typically, a script will be designed to integrate with a business application such as Oracle Customer Support (a component of Oracle TeleService, and formerly known as Oracle Customer Care) or Oracle TeleSales. The CRM business application performs most of the "contact management" and provides all essential data collection and reporting; the Oracle Scripting component gives the agent additional information that makes it easier to establish a dialog with the customer by ensuring that the agent has a clearly guided presentation.

As of release 11.5.2, Oracle Scripting integrates with two applications mentioned above (TeleService and TeleSales) within the Oracle eBusiness Suite. Integration in release 11.5.3 and beyond is expected to expand to other Oracle applications in the CRM application space. For more information on Scripting integration, see Scripting and Integration of Other CRM Applications in this document.

Scripting Typically Requires Customization

Unlike other CRM applications, Oracle Scripting requires customization (in the form of a script tailored to the enterprise's needs) in order for the implementation to be of use to the enterprise. Each customer has a unique set of requirements to meet their needs for each script. Based on the complexity of the script to be delivered, this is likely to be a non-trivial effort. Typically, at least one customized script is expected to be delivered by OCS or consulting partners, either as part of the implementation process or as a phase immediately following implementation.

While it is conceivable that the implementing enterprise could develop its own scripts with the appropriate knowledge and training, is highly recommended that the initial script be put into production by OCS or experienced consulting partners. This approach will most likely be more cost-effective and certainly more expedient. With training, and with a functional script in production in an interaction center from which to model, an enterprise is more likely to experience success modifying existing scripts or creating new ones for emerging business cases.

Management team members of both the implementing enterprise and the Oracle-affiliated consulting team should be aware of the requirement to create a customized script at the outset. Any suitable implementation methodology will include, as a part of the discovery and requirements gathering process, the gathering of business rules and a scrutiny of the particular campaign or business case in order to develop a script that meets the enterprise's needs. This means that the implementation process for Oracle Scripting usually consists not only of installation and implementation of the product, but also ensuring that the customized script(s) are available and functional.

Scripting Provides for Alternative Graphic User Interfaces

Although the Java-based GUI shown in Figure 1 is the default GUI for the Scripting Engine, Oracle Scripting was designed to allow for customizing all or part of the graphic user interface by manipulating the Java methods in the `iesclien.jar` file. Obviously this is considered customization of the application, and any enterprise considering this should proceed under the caveat that customized applications are only supported to a limited degree by OSS, as based on the standard Support agreement. Customization of the presentation GUI for the Scripting Engine is not within the scope of this document.

Scripting Operates in Multiple Architectures

Functionally, Oracle Scripting supports more than one architectural model that distributes processing across the network in multiple tiers. Scripting currently supports a two-tiered architecture (referred to as the "Caching Architecture"), and a three-tiered architecture (referred to as "3-Tier JServer Technology Stack Architecture").

This document discusses implementation criteria supporting both architectures. From an implementation perspective, when procedures or approaches differ, these differences will be indicated in this document.

However, Oracle Corporation *strongly recommends* implementing Oracle Scripting using the Caching Architecture, as *only Caching Architecture implementations will be*

supported in the near future. For more information, see Application Architecture > Caching Architecture Versus Three-Tier JServer Technology Stack Architecture > Support Statement below.

Application Architecture

Java Technology—The Script Author

The Script Author is a 100% Java application. When the Script Author is launched, the script development tool appears in a Java-based GUI. Using this GUI, scripts can be created or modified, and deployed from the Script Author workstation to the Oracle Applications database, providing for a 2-tier architecture (client to database).

The Script Author is installed on a Windows NT, 95, 98 or 2000 client (typically a development workstation) by the Oracle Universal Installer.

Java Technology—The Scripting Engine

The Scripting Engine is a 100% Java software-based engine that interprets business rules and drives the flow of the script.

The Scripting Engine classes are stored in the Scripting foundation JAR files, and are interpreted by the Java Virtual Machine (JVM). The Scripting Engine interprets scripts deployed in the database, monitoring the agents' progress from panel to panel in the script, and drives the flow of the script based on a combination of:

1. The business rules created by the various branch types connecting objects on the canvas in the Script Author document,
2. The agent's responses entered into the Script Engine GUI using various answer controls (buttons, check boxes, text fields and areas), and
3. Dynamic evaluation of Commands programmed in the script (Java, PL/SQL, Forms, Blackboard, and Constant commands).

Commands are referenced in the script by creating Actions in the Script Author, which are associated with objects on the canvas such as panels, blocks, groups, branches, or even the global script itself. Some commands may reference associated custom code, for example custom Java methods deployed to the server, or custom PL/SQL procedures or packages stored in the database.

Where Is the Scripting Engine?

The Scripting Engine executes the business logic in the JVM. For Caching Architecture implementations, the JVM is provided by JInitiator on the agent

desktop (the client tier). For 3-tier JServer Technology Stack implementations, the JServer provides the JVM, executing business logic on the applications tier of the server.

In the Caching Architecture, when an agent requests a particular script to be launched from Oracle Applications, the database server is contacted using the thin JDBC protocol, and the script (and any custom code) is cached to the client tier to execute. The first time the script is run, a delay may be encountered while files are caching, although in second and subsequent script sessions, providing an updated script has not been deployed to the database, the cached files are used, speeding operations. When the script is updated, a fresh copy will be downloaded and cached to the client.

Using the 3-tier JServer Technology Stack architecture, when an agent requests a particular script to be launched from Oracle Applications, the client uses IIOP to contact the application server (JServer). The JServer, in turn, contacts the database server using thin JDBC, and the script (and any custom code) is cached to the applications server to execute.

Only Caching Architecture implementations are recommended at this time. For more information, see Caching Architecture Versus Three-Tier JServer Technology Stack Architecture below.

Integration With Forms Architecture

Although Scripting is not built with Oracle Forms, scripts can integrate with Forms-based applications. Some *limited* out-of-the-box integrations with CRM applications currently exist, and more are expected in the future. For more information on Scripting integration, see Scripting and Integration of Other CRM Applications in this document.

Caching Architecture Execution

The presentation layer is executed on the client in both Caching Architecture and 3-tier JServer Technology Stack models. In other words, interaction center agents view a script in runtime on their workstations. For Caching Architecture implementations, the execution of the business logic is *also* executed on the client tier (the agent workstation).

The second tier is the data tier, represented by the Oracle 8*i* database. This is the database installed during the Rapid Install and used by all CRM and ERP applications; it is often referred to as the “applications database.“ The data tier includes (if necessary) other custom databases and legacy systems that contain data needed by the script.

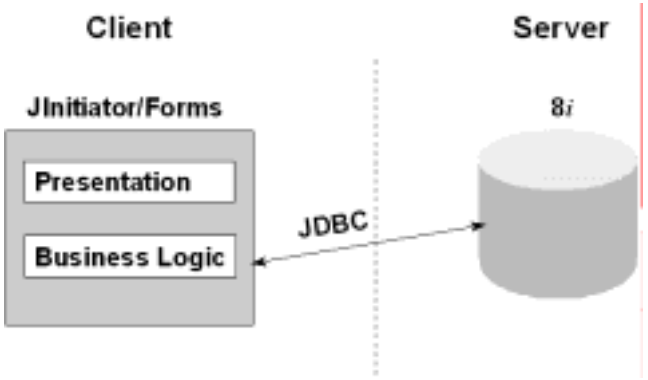


Figure 4. Caching Architecture of Oracle Scripting.

Caching Architecture Example

Agents log into Oracle Applications with a responsibility that is associated with Oracle Scripting (and other applications, as appropriate). When the agent invokes a script for the first time, the script is downloaded from the Oracle 8*i* database (using thin JDBC as a protocol), and the script (and any associated Java) is cached from the data tier to the agent workstation. Oracle JInitiator provides the JVM on the client, which interprets the Java code (base Scripting functionality *and* custom Java) and serves as the Scripting Engine on the agent desktop.

Oracle JInitiator may be downloaded when an agent logs into Oracle Applications. For more information about Oracle JInitiator, see Related Products and Components > Oracle JInitiator.

Increased Agent Desktop Requirements

Because more work is done on the client, the agent may need a more powerful agent desktop for a Caching Architecture implementation. For example, at one large implementation running a complex script, agents use Pentium 2 CPUs running at 450 MHz with 128 MB RAM. The desktop requirements depend in large part on the complexity of the script.

Three-Tier JServer Technology Stack Execution

The architecture is different for 3-tier JServer Technology Stack implementations, with dramatically different repercussions. The Java (both for the Oracle Scripting application and any custom code associated with a particular script), as well as the requested script, are stored in the data tier (described above).

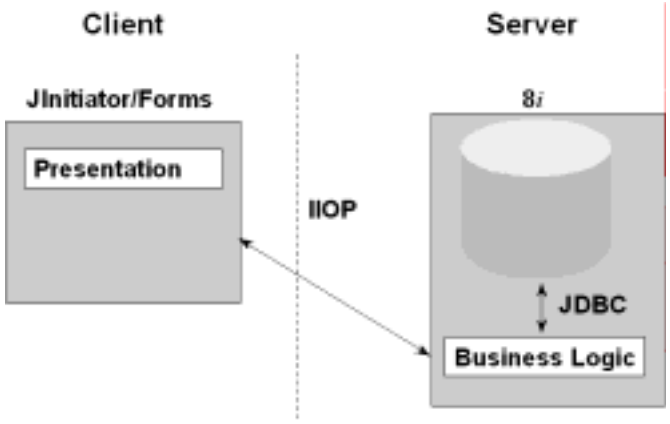


Figure 5. Three-tiered JServer technology architecture of Oracle Scripting.

Oracle 8*i* JServer provides the JVM as a middle tier, which interprets Java on the server, including the Scripting foundation JAR files that provide the Scripting Engine. The Scripting Engine interprets the script, and any appropriately deployed custom Java code, in the middle tier. In contrast to the Caching Architecture model, the script is never downloaded to the client in 3-tier JServer Technology Stack operations, and all business logic is executed on the applications server rather than on the client workstation.

As always, the presentation layer is executed on the client tier.

Three-Tier JServer Technology Stack Example

Agents still log into Oracle Applications with a responsibility associated with Oracle Scripting (and other applications, as appropriate). Each time the agent requests a script, the IIOP protocol is used to invoke the Scripting Engine to run the script from the Oracle 8*i* database.

In the 3-tier model, the agent is then required to log in again using a distinct username and password established for Scripting users. Once the agent login is authenticated in the database, the Scripting Engine launches the requested script, using the JVM (and therefore, the processing power) of the applications server to execute the business logic. The GUI or presentation logic is executed on the client tier (the agent workstation).

Three-Tier JServer Technology Stack Example

Agents still log into Oracle Applications with a responsibility associated with Oracle Scripting (and other applications, as appropriate). Each time the agent requests a script, the IIOP protocol is used to invoke the Scripting Engine to run the script from the Oracle 8*i* database.

In the 3-tier model, the agent is then required to log in again using a distinct username and password established for Scripting users. Once the agent login is authenticated in the database, the Scripting Engine launches the requested script, using the JVM (and therefore, the processing power) of the applications server to execute the business logic. The GUI or presentation logic is executed on the client tier (the agent workstation).

Increased Implementation Requirements

Creating additional agent logins for each agent in the interaction center requires repetitive manual steps to be performed (once per agent), and necessitates detailed coordination. Essentially, the work required specifically to manage users is doubled from that of Caching Architecture implementations, since separate Oracle Forms/Applications logins are still required. Additionally, as turnover is traditionally fairly high in interaction centers, providing new scripting logins for incoming personnel creates an additional burden of maintenance on interaction center managers or administrators.

Java Technology—Oracle 8*i* JServer

JServer is a new technology that first became available with the initial release of Oracle 8*i*. The Oracle 8*i* JServer hosts and runs Java objects in a special Java Virtual Machine (JVM) built into the database.

JServer is built into the Oracle 8*i* database. When executing in 3-tier JServer Technology Stack environments, Scripting uses the CORBA Object Request Broker (ORB) for client-to-middle-tier communication.

JServer includes a Java Database Connectivity (JDBC) driver that bypasses the Oracle Call Interface (OCI) and network layers, thereby making it more efficient.

Sessions and Three-Tier JServer Technology Stack Execution

Internet Inter-ORB Protocol (IIOP) is the protocol used to initiate a CORBA session. Any time a script is launched in the 3-tier JServer technology stack, the server proxy bean instantiates a CORBA session on both the client (agent desktop) and the JServer. These sessions communicate with the Oracle 8*i* database via the special native JDBC driver, and with other databases and systems via JDBC or other protocols. All communication is performed through that session. For more

information, refer to the *Oracle 8i Enterprise JavaBeans and CORBA Developer's Guid*e, Chapter 4—Connections and Security.

Caching Architecture Versus Three-Tier JServer Technology Stack Architecture

Support Statement

1. Oracle Corporation *highly recommends* implementing Oracle Scripting using the Caching Architecture.
2. *All customers* currently using Scripting on 3-tier JServer Technology Stack architecture are advised to *migrate to a Caching Architecture* by April 1, 2001.
3. As of April 1, 2001, Scripting 3-tier JServer Technology Stack implementations *will no longer be supported.*

For information on how to move from a 3-tier JServer Technology Stack implementation to a Caching Architecture model, see Converting Three-Tier JServer Technology Stack Environments to a Caching Architecture below.

Scalability

Scalability issues have been identified with the 3-tier JServer technology stack architecture in large-scale implementations of Oracle Scripting. Performance testing indicates that responsiveness and overall system performance may be at risk for any large 3-tier JServer Technology Stack implementation.

Further testing indicates that we can successfully scale large, complex scripts to many agents in a large production call center (with, for example, 700 concurrent agents) using the Caching Architecture.

For these reasons, Oracle Corporation has decided to support only the Caching Architecture, as less risk is imposed on customers following this model.

For information on how to move from a 3-tier JServer Technology Stack implementation to a Caching Architecture model, see Converting Three-Tier JServer Technology Stack Environments to a Caching Architecture below.

Scripting and Port Numbers

The default database port number for the two-task common (TTC) layer is 1521. The default database port number for IIOP is 2481.

TTC is the service that processes incoming Net8 requests for database SQL services from Oracle tools (such as SQL*Plus), and customer-written applications (using Forms, the OCI layer, etc.). Tools such as LoadJava and DropJava connect using a TTC port; when deploying a script to the applications database from the Script

Author, you must use the TTC port. IIOP is the protocol used to initiate a CORBA session. This is required for 3-tier JServer Technology Stack execution.

For More Architecture Information

Technical documentation regarding Scripting, its architecture, Application Program Interfaces (API), and other material can be found in various documents listed in the Related Documentation and Resources section in this document.

Using Oracle Scripting

Overview

Within the context of a Scripting implementation, a script must be opened in the Script Author, deployed to the applications database, and successfully launched and executed in the Scripting Engine to ensure proper function. This section, as well as the practice exercises included in Appendix B, address the basics of using Oracle Scripting. For more information on using Scripting, including detailed discussion of the tools, commands, and procedures to create functionality in the script and execute that functionality in runtime, refer to the *Oracle Scripting Concepts and Procedures* document.

Note: It is not necessary to *create* a script from scratch in order to confirm a successful installation and implementation, as long as you have access in the target environment to scripts *known to function in a tested environment of the same architecture.*

It is recommended to test, in that environment, at least one pre-tested script with *no* custom Java methods, as well as at least one test script *with* custom Java methods, such as the Regression Test Script discussed below. For more information, see Testing Oracle Scripting below.

Using the Script Author

As discussed in Understanding Oracle Scripting above, Scripting consists of two components: the Script Author (a development tool to build scripts) and the Scripting Engine (a runtime engine to present those scripts in a thin-client agent desktop application).

The Script Author is a development tool that allows script authors to use graphical tools to create and modify scripts. The entire script is represented as one or more graphs that use visual symbols to represent the three main object types— individual panels, groups of panels, and code blocks—connected with the appropriate branch type to control the flow of the script. Each valid script will also contain a start node (created on each graph or sub-graph automatically) and a terminal node which must be placed on the canvas by the author of the script. Figure 6 shows some of the basic components of the Script Author development environment, although for an exhaustive list of all of the various icons and how to use them, refer to the *Oracle Scripting Concepts and Procedures* manual.

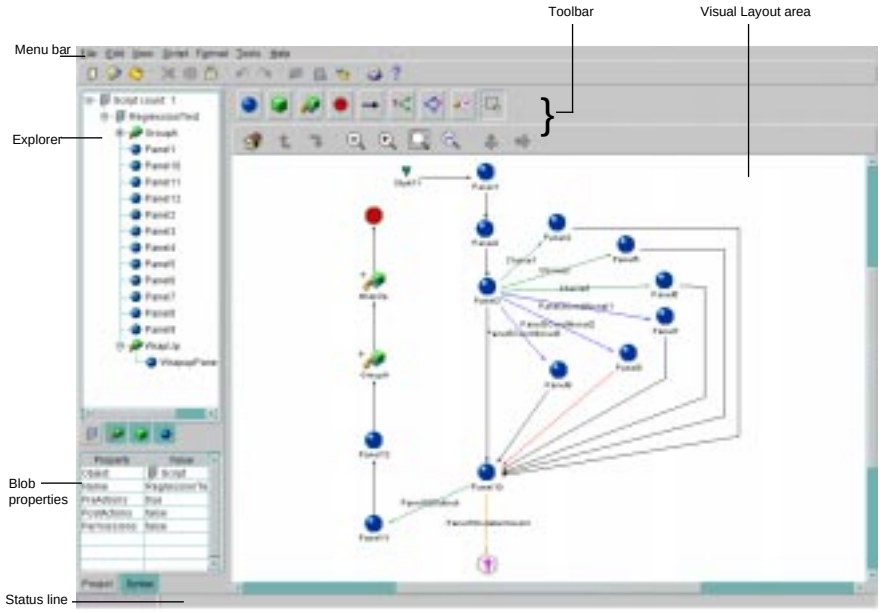


Figure 6. The Script Author development environment

Creating and Modifying a Script

The step-by-step practice exercises described in Appendix B describe how to:

- Create a syntactically correct script using the Script Author component of Oracle Scripting
- Syntax check, compile, and deploy a script to the applications database
- Modify and add to an existing script
- Create distinct branching and multiple Lookup Values
- Enable the Disconnect feature within a script by creating the WrapUpShortcut group
- Use Commands where Java is not required

Requirements for Script Development

Individuals authoring and testing scripts for an interaction center will need a networked script development workstation (Windows NT, 95, 98, or 2000) loaded with the Script Author and an Oracle CRM 11*i*-compliant Web browser. Oracle JInitiator is also required, although is not prerequisite because the correct for the appropriate environment may be downloaded when an agent logs into Oracle Applications at runtime. For more information about Oracle JInitiator, see Related Products and Components > Oracle JInitiator.

From a network perspective, this workstation must be on a network and able the applications database server in order to deploy scripts.

Other Considerations for Development Workstations

If roles are divided for Scripting development environments between those developing strictly in the Script Author and those developing custom code such as PL/SQL, Java, and Forms, the Script Author development workstation has no other software requirements; no other Oracle Applications are required to be installed on the development machine. The Script Author uses thin JDBC to deploy scripts to the database, bypassing the OCI layer and therefore obviating even the requirement for appropriately configured `TNSNAMES.ORA` and `SQLNET.ORA` files.

However, if the intention within the implementing enterprise or Consulting home base is to have one or more primary development workstations where all script development and testing will take place, these development workstations would benefit from appropriately configured `TNSNAMES.ORA` and `SQLNET.ORA` files. The presence of Oracle Client (including tools to allow the developer to make SQL calls

to the database), a Java IDE with appropriately configured libraries, and other helpful tools, are recommended for a fully capable development workstation. See the section Using Oracle Scripting > Writing and Deploying Custom Code to Enhance a Script for details on additional requirements for a development workstation.

Creating Simple Scripts

If you need to create scripts, refer to Appendix B and follow the procedures described in detail in Exercises 1 through 5.

Creating a Complex Script

Scripting is predominantly a single-developer tool. Early in development stages, multiple developers can theoretically work on different scripting portions of the script in the Author environment if the functionality of those portions is distinct. While Oracle Scripting has no team development tools in the R1 or R2 releases, it does allow development teams to set up different "groups" (an example of such is the WrapUpShortcut group described in detail in Exercise 3 of Appendix B). These groups must contain discrete functions within the script as a whole. Each Script developer can work on a specific group, or set of groups, and then combine those groups to create an integrated script.

Challenges to Scripts Built in Parallel with Multiple Developers

Note that, once certain groups or functions begin to call other groups or functions, the Scripting groups must be carefully integrated in order to syntax check the global script. This is also required in order to compile and/or deploy the script for testing purposes. This complicates development and increases the burden of communication on all team members. Parallel development is only recommended in cases where project management is experienced, and has the opportunity for frequent, detailed communication with team members on a daily (and sometimes hourly) basis. In such instances, careful, frequent backing up of individual groups (and tested, functioning integrated scripts) is an *absolute requirement.* Note that this may require substantial hard drive or network storage for the duration of the project.

Benefits and Challenges to Multiple Development Roles

Multiple roles in development can further the early progress of a project. For example, one specialist can develop database hooks, another can create PL/SQL packages, a third can work primarily on custom Java methods, a fourth can focus on creating primarily sequential call guide panels, etc. However, the same caveats

voiced above hold true in terms of integration or testing for each of these components.

Integrating Database Calls to a Script In Development

In regard to integrating database components, additional challenges result. In isolation, database components within a script can be developed and even tested, but when integrated into the main script, the development process is greatly complicated. From that point forth, all developers require a live database connection to continue development and testing of any subsequent portion of the script, and any changes made to database hooks by that specialist must be carefully integrated (and in fact thoroughly tested) before development can continue.

In such instances, it is recommended that development be allowed to proceed prior to final integration of database calls for as long as possible, to eliminate the additional time constraints on development personnel. Another approach is that a "dummy path" can be introduced into the script whereby most groups can be reached without accessing the panels or blocks that contain database procedures.

Unless such measures are taken, once integration of database calls has been completed, parallel development must cease, and a sole developer must work on the script at a time, increasing the amount of time required for script completion, increasing the burden on a single developer, and complicating further development and testing of the script.

Summary

With careful management, individual developers' portions can be extricated from a tested script for continued development, but constant communication, testing, backing up, etc. are required if this approach is taken.

Compiling and Deploying a Script

Oracle Scripting Script Author has a built-in syntax-checking feature, available by selecting “Syntax Check“ from the **Tools** menu or clicking the “Check Syntax“ button in the tool bar. When you deploy a script (by selecting “Deploy Script“ from the **Tools** menu or clicking the “Deploy Script to DataBase“ button in the tool bar), the Script Author first automatically checks the syntax of the script, then (if successful) compiles the script and finally deploys the script to the applications database. The following parameters are required for deploying a script:

Field Type	Control	Value
Existing Connections:	Checkbox	Do NOT click
Reuse existing connection:	Drop down list box	Do NOT click
Connection Name:	Text field	This field is ignored
JDBC Driver:	Text field	oracle.jdbc.driver.OracleDriver
URL:	Text field	jdbc:oracle:thin:@<server machine name>:<TTC port number>:<SID>
User Name:	Text field	<apps user DB login>
Password:		<apps user DB password>

Deployment Parameter Notes:

JDBC Driver. Note that the JDBC driver field is case-sensitive.

URL. The following apply:

- Based on the workstation’s environment (e.g., TNSNAMES.ORA and SQLNET.ORA configurations), the server machine name variable may require a full URL (such as server.us.oracle.com). Follow the same procedure for that environment that you would use to run SQL*Plus or SQL Worksheet to connect successfully in that environment to the applications database.
- You may need to consult with a DBA regarding the value for the TTC database port number, although this is typically 1521. When a DBA sets up a database instance, she sets up a database port for that instance. Thus, common ports

include 1521, 1522 or 1523, and so on. Note that this is distinct from the IIOP port number. See Scripting and Port Numbers in this document.

- The DBA will also have the SID information you need to connect.

User Name and Password. Required here is the `apps` (applications) username and password.

The Apps User

The `apps` user is a special database account which is central to all applications. Every schema object that is created by an Oracle Application is available in the `apps` schema (that is, every object is viewable if you are logged in as `apps`). In addition, regardless of which `apps` account you use to log in to Forms-based Oracle Applications (for example, if you log into Oracle Applications as `sysadmin`/`sysadmin`), you actually connect to the *database* as `apps`. The `apps` username and password is automatically granted privileges to deploy scripts at the DBA or systems administrator level.

Writing and Deploying Custom Code to Enhance a Script

Scripts can be customized in the Script Author by referencing custom code from the Script Author. The custom code is not generated by Scripting—it must be created by a qualified expert, typically deployed to the server, and referenced from the script.

Custom Code and Scripting Architecture

The architecture type employed in a Scripting implementation is relevant to which steps are required for deploying custom code. Oracle Scripting currently supports the 3-tier JServer model as well as the Caching Architecture. While the Caching Architecture is recommended (and the 3-tier JServer Technology Stack model is not expected to be supported following April 1, 2001), differences for implementing and maintaining custom code for both architecture types are documented herein.

Commands Supported by Oracle Scripting

Scripting supports Java, PL/SQL, Blackboard, Oracle Forms, and Constant commands.

Blackboard commands are unique to Scripting. They reference specific Scripting APIs built into the product, and can be defined from within the Script Author. (Blackboard APIs are also available to be used with custom Java code; see the published Scripting APIs document.)

Constant commands can also be defined from within the Script Author. They return a single, specified value (defined from within the Script Author). This constant value can be called upon at runtime in the Scripting Engine, either to display the value or to be passed as a parameter to be used in another command.

The remaining commands typically take custom code that are referenced from within the script using the Script Author, but exists elsewhere.

Actions Reference Commands

For the most part, custom code is called by a Command using Pre-Actions, Post-Actions, and Actions.

Actions are containers for commands that are associated with branches of any type. **Pre-** and **Post-Actions** are containers for commands for specific panels, groups, blocks, or the global script.

Actions

As the name suggests, the command associated with an **Action** occurs *concurrently with* the progression or flow of the branch type in the script.

Pre-Actions

Commands associated with Pre**-Actions** occur *prior to* execution of the code represented by the scripting element that holds the Pre-Action. For example, a command associated with a Pre-Action of a panel is executed immediately prior to the display of panel text and answer controls associated with that panel. As a second example, a command associated as a Pre-Action to the global script will be executed *upon instantiation of the script*, and *prior to* display or execution of the first functional object on the Script Author canvas.

Post-Actions

Conversely, commands associated with Post**-Actions** occur *immediately following* the execution of the code represented by the scripting element that holds that Post-Action, and *prior to* the progression of the script from that object on the canvas to the next. A Post-Action of the global script is a good place for housekeeping, performing PL/SQL commands, writing information (using Java Commands) from the Scripting Blackboard to the database, etc.

In current releases, the Command dialog box looks identical for any type of Action within the Script Author. Once you select the type of Action, parameters specific to that type of Action can be entered.

For details on how to define and configure Commands, see the *Oracle Scripting Concepts and Procedures* document.

PL/SQL Code and Queries

Scripts can be customized in the Script Author with the use of SQL Commands, which call custom PL/SQL code from the database during runtime. The Script Author does not provide any facilities internally to write, compile, test, or deploy custom PL/SQL code. It is expected that this task is performed by an experienced PL/SQL developer, and that the custom PL/SQL code is developed and tested in the database before the script is configured to call it.

There is no special step needed to deploy PL/SQL code in order for Scripting to access it. As long as the PL/SQL is in the database, it can be accessed during script execution.

Developers writing PL/SQL code will need to have a networked development workstation (Windows NT, 95, 98, or 2000) appropriately configured (including `TNSNAMES.ORA` and `SQLNET.ORA` settings) to connect to the database and with SQL tools such as SQL*Plus or SQL Worksheet (available with the installation of Oracle Client). Additionally, the Script Author should be loaded, and an Oracle CRM 11*i*-compliant Web browser. In this way, SQL commands intended to be executed from Scripting can be verified in advance using any SQL tool, or tested by running the Scripting Engine through use of the Demo Form from the development workstation. JInitiator is also required (but may be downloaded at runtime). For more information about Oracle JInitiator, see Related Products and Components > Oracle JInitiator.

For more details on how to define and configure SQL Commands, see the *Oracle Scripting Concepts and Procedure*s document or an appropriate SQL reference such as the *Oracle 8i SQL Reference* document.

Java Code

Scripts can be customized in the Script Author with the use of Java Commands, which call custom Java methods from the server during runtime. Custom Java code is required to manipulate Scripting functionality via the published Java APIs. The Scripting API is encapsulated in the Proxy Bean object. The Scripting Engine includes the ability to pass a Proxy Bean object as a parameter to a Java Command object so that the custom code can call Scripting APIs.

Custom Java methods created to add specific functionality to the script are typically written by an experienced Java developer, tested, compiled, and compressed into a Java archive (JAR) file. Each Java method must then referenced in the Script Author

by attaching a Java Command (typically, as one of three types of Actions described above) at each point in the script that the code is to be invoked at runtime.

Deploying Customized Scripts

Scripts containing references to each custom Java method are deployed to the database from the Script Author in the usual fashion (**Tools** > **Deploy Script**). The modified script is deployed from the authoring tool using thin JDBC. Each script is stored in the Oracle 8*i* database schema in the IES_ DEPLOYED_SCRIPTS table.

Deploying Custom Code

The custom Java code (in the form of a `.jar`, `.class`, or `.zip` file) must also be loaded and stored on the server. The method used to deploy the custom code, and the location on the server in which the code is deployed, differs by the type of architecture supported.

For Caching Architecture environments, this code must reside in the APPL_TOP as described in Deploying Custom Code to the APPL_TOP below. For 3-tier JServer Technology Stack environments, custom code is loaded to and stored in the JServer (as described in Executing LoadJava and DropJava below) so that it can be accessed and run in the JServer during script execution.

The Script Author does not provide any facilities internally to write, compile, test, or deploy custom Java code. It is expected that this task is performed by an experience Java developer, with access to a full Java Integrated Development Environment (IDE) that is Java Development Kit (JDK) 1.1.8-compliant and has full editing, compiling, and debugging facilities. Developers writing custom Java code will need to have a networked development workstation (Windows NT, 95, 98, or 2000) loaded with the Script Author and an Oracle CRM 11*i*-compliant Web browser. The Java IDE may require libraries to be added manually in order to compile Scripting-specific Java methods. For more information, see JAR Files and the Java IDE below.

For more details on how to define and configure Java Commands, see the Oracle Scripting Concepts and Procedures document.

JAR Files and the Java IDE

Depending on the Java integrated development environment used to develop custom Java, it may be required to add the foundation Scripting JAR files to the IDE as libraries. For example, JDeveloper requires two of the three foundation JAR files, `iesclien.jar` and `iescommn.jar`, to be resident on the development workstation and referenced as libraries. These will have to be obtained from the applications tier where they are installed during the Rapid Install, and copied to

each Java development workstation. Based on which version of JDeveloper, it may be necessary to configure the IDE both for the JDeveloper application as a whole and for specific projects. For information on how to add libraries to Oracle JDeveloper, refer to that product's documentation.

Adding Import Statements to Java IDE

In order to compile code that calls the Scripting APIs, the following import statements will be needed:

```
import oracle.apps.ies.integration.common.*;
```

```
import oracle.apps.ies.integration.common.exception.*;
```

These classes are in the `iescommn.jar` file. The correct version of this JAR file (the one that will be updated when a patch is run) is the one on the web tier that is downloaded to the client by JInitiator. This file is in the Apps file system (known as APPL_TOP) under the following directory: `$JAVA_TOP/oracle/apps/ies/jar`

Always make sure you have a copy of the latest version of this JAR file in your classpath on the machine that is being used to compile your custom code.

Deploying Custom Code to the APPL_TOP

Caution: ***Do not perform these steps for 3-tier JServer Technology Stack implementations.*** Deploying custom Java JAR files is appropriate only for Caching Architecture implementations.

For Caching Architecture operations, custom code must be deployed to the APPL_TOP and referenced in the `appsweb.cfg` file.

Follow the steps indicated below, using a sample filename of **`custom.jar`** to represent the custom code.

Checklist for Deploying Custom Java Code to APPL_TOP

> **Note:** The following checklist steps are described in further detail in the following section.

1. Compile and compress custom code into a JAR file.
2. Verify existence of (or create) `$JAVA_TOP/ies_custom` directory.
3. Deploy compiled and compressed custom code to `ies_custom` directory.
4. Append filename and path of custom JAR files to **`appsweb.cfg`** file.

Detailed Steps for Deploying Custom Java Code to APPL_TOP

> **Caution:** The following steps require knowledge of patching and installing Oracle Applications, and requires knowledge of the APPL_TOP and JAVA_TOP, as well as knowledge of the `appsweb.cfg` file and how it is used. Prior to modifying the `appsweb.cfg` file, *ensure a backup copy is retained.*

> **Caution:** This is a detailed description of the steps listed briefly in the Checklist for Deploying Custom Java Code to APPL_TOP above. *It is not intended for this full list of steps to be repeated.*

1. Compile and compress custom code into a JAR file.

   Compile any custom Java code into `.class` files. Then compress the compiled classes into `.jar` files. Note that many `.class` files can be combined into a single `.jar` file. For the sake of simplicity, this is recommended. In any case, ensure all compiled classes are converted to `.jar` files. For this example, assume we have combined all compiled custom Java classes into a Java Archive called `custom.jar`.

2. Verify existence of (or create) `$JAVA_TOP/ies_custom` directory.

   Check the Applications server to determine if an `ies_custom` directory exists as a child to the JAVA_TOP (a subdirectory of the APPL_TOP). This is where

custom JAR files will be loaded for Caching Architecture operations. If this directory does not already exist, create:

```
$JAVA_TOP/ies_custom/
```

3. Deploy compiled and compressed custom code to `ies_custom` directory.

   Move the newly created `.jar` file(s) to the `ies_custom` directory on the JAVA_TOP, through whatever method available (i.e., FTP, or if you have direct access to the server, mount drives as necessary and copy custom JAR files over). For our example, the end result would be one new JAR file (`custom.jar`) located in the directory created in the previous step:

   ```
   $JAVA_TOP/ies_custom/custom.jar
   ```

4. Append filename and path of custom JAR files to **`appsweb.cfg`** file.

   Append the path and name of the custom JAR file (in this example, `OA_JAVA/ies_custom/custom.jar`) to **`appsweb.cfg`** file. Custom JAR files must be referenced in the configuration file to *precede the reference to* `fndlist.jar`. All files that *follow* the `fndlist.jar`, regardless of whether they are in the same archive or in later archive blocks, are read as on-demand files. For more information, *see the note below.* For an example of the updated **`appsweb.cfg`** file, see Appendix D.

Note: Step 4 is the last step in the process of deploying custom Java code to the APPL_TOP for Caching Architecture configurations. However, other modifications to the **`appsweb.cfg`** file will be required. See Caching Architecture Implementation Checklist for new implementations, or Converting Three-Tier Environments to a Caching Architecture if that situation is relevant.

Executing LoadJava and DropJava

Caution: ***Do not perform these steps for Caching Architecture implementations.*** Loading or Dropping Java to the JServer (with or without `-grant`) is appropriate only for 3-tier JServer Technology Stack implementations.

When to Use LoadJava

For 3-tier JServer Technology Stack operations, scripts developed in the Script Author are deployed from the authoring tool (using thin JDBC), which stores each script in the Oracle 8*i* database schema in the IES_ DEPLOYED_SCRIPTS table. Custom Java classes referenced by Commands in the Script Author—and any third party libraries that the custom code calls—must be loaded into the 8*i* JServer. This includes the Java methods that support Scripting's base functionality. These files—typically `.zip` or `.jar` files, although they may be decompressed `.class` files—are loaded into the 8*i* JServer using LoadJava.

Custom Java must be loaded using the `-grant` option. This option explicitly allows use of those Java methods by the specified agents. If this option is excluded from a LoadJava command, only the `apps` user agents (a superuser with `apps` privileges to the database) will be able to run scripts with that custom code.

The Rule That Proves the Exception

When implementing in the 3-tier JServer Technology Stack, two of the three main Java classes that support Scripting— `iesservr.jar` and `iescommn.jar`—are deployed to the database *twice*. The first time they are loaded with LoadJava, agents have not yet been created. Consequently, at that time, LoadJava is used *without* the `-grant` option. Note, however, that these classes are dropped from the JServer (using DropJava) and then redeployed using a LoadJava with `-grant`.

LoadJava Syntax

```
loadjava -grant <agent_name> -user apps/apps@<SID> -resolve
-oracleresolver -synonym -definer -oci8 <jar file being
loaded>
```

Example:

```
loadjava -grant ies_agent -user apps/apps@iem11icw -resolve
-oracleresolver -synonym -definer -oci8 cstmjava.jar
```

Perform as many times as necessary, once per agent. Alternatively, a database script can be created to include the LoadJava command multiple times, each with an explicit `-grant` for every specific agent in the interaction center.

When to Use DropJava

For 3-tier JServer Technology Stack operations, any time custom Java is changed, the compiled customized Java code (`.jar` or `.class` files) must be *reloaded* into the JServer using LoadJava.

Also, when the base product (Oracle Scripting) is upgraded, replacing any of the three foundation Java classes), the compiled foundation Java code (`.jar` files) must be *reloaded* into the JServer using LoadJava.

Prior to reloading custom code or upgraded product foundation Java code, perform a DropJava specifically referencing that compiled file.

Note: Executing DropJava is not required the first time custom Java code is loaded into the JServer.

DropJava removes the specified files from the JServer for all agents. Unlike LoadJava, t*his need not be performed for each agent.*

Exercise Due Caution. If you do not specify which files to be dropped when DropJava, all Java classes will be deleted from the JServer. Be sure to specify which files to drop. *Do not use* `-grant` *option with DropJava.*

DropJava Syntax

```
dropjava -user apps/apps@<SID> -resolve -oracleresolver
-synonym -definer -oci8 <jar file being dropped>
```

Example:

```
dropjava -user apps/apps@<SID> -resolve -oracleresolver
-synonym -definer -oci8 cstmjava.jar
```

Verbose Command for LoadJava and DropJava

To troubleshoot the use of the LoadJava and DropJava utilities, add the `-verbose` option to the command line arguments. This will cause tool debugging output to be printed to the console.

Types of Files Used in Developing a Script

The typical script development process involves at least 4 different types of files:

- **`.script` file—**Contains script metadata. This is the file format created by the Script Author. The `.script` file is deployed to the database from the Script Author for execution in the Scripting Engine using both the Caching Architecture and the 3-tier JServer architecture.

- **`.java` file—**Contains Java methods intended to add functionality to the desktop application. The Java methods contained in these Java files are needed for scripts to execute custom code, to evaluate complex logic, or to redirect the flow of a script based on specified parameters. For example, any indeterminate branch in a script will have a Command that references a Java method. The `.java` file is a Java source file where Java methods are created in the developer's Java IDE of choice. It is from this file that compiled `.class` files are generated. The `.java` source files are *not* typically deployed to the server, as only the compiled file is required.
- **`.class` file—**Contains compiled Java methods that are referenced by the `.script` file. The `.class` file(s) are associated from the Commands dialog within the Script Author. For Caching Architecture implementations, `.class` files *must* be combined into `.JAR` files (described below). For 3-tier JServer Technology Stack environments, `.class` files *can* be deployed to the JServer, although deploying `.JAR` files is *always recommended*.
- **`.jar` file—**Compressed Java Archive files. This format must be used for Caching Architecture environments. Creating a `.JAR` file from one or more compiled `.class` files can be achieved using the JAR command (part of the JDK) or by utilities within a Java IDE for that purpose.

What Code Must Be Deployed?

1. *Any time a **script** is created or modified* using the Script Author, it must be deployed to the applications database from within the Script Author as described in Compiling and Deploying a Script section above.
2. *Any time **custom Java*** referenced by a script is *created* or *modified*, it too must be deployed to the server. The destination point of that custom code depends on the architecture used. For details, see Writing and Deploying Custom Code to Enhance a Script > Deploying Custom Code above.
   - The Java IDE typically retains uncompiled code as `.java` files. Compiling Java creates smaller `.class` files. Whether your custom code consists of one or several `.class` file(s), these can be further reduced in size by compressing them into one (or more) Java archives or `.JAR` files.
   - For 3-tier JServer Technology Stack implementations, it is recommended (for sizing and space considerations as well as performance) that you limit the size and amount of files sent to the database. Even though 3-tier JServer Technology Stack operations supports executing custom Java code from both `.jar` files and `.class` files, it is recommended to always use the smaller `.jar` file format.

- The `.java` source files (and larger-sized compiled `.class` files) are not required to be deployed. The `.java` source file should be retained by the developer (the associated `.class` file can always be regenerated), but should not be deployed to the JServer. Otherwise, these files will consume much-needed database storage space, which may cause runtime errors.

3. Any time a PL/SQL command or package is referenced in a script, that PL/SQL code must be deployed to the database. Consult a DBA for more information.

Using the Scripting Engine

As discussed in Understanding Oracle Scripting above, Scripting consists of two components: the Script Author (a development tool to build scripts) and the Scripting Engine (a runtime engine to present those scripts in a thin-client agent desktop application).

The Scripting Engine executes the GUI on the agent desktop, stepping the agent through the script and determining the script flow dynamically based on the agent's response and the business rules built into the script.

JAR Files and the Scripting Engine

Oracle Scripting makes use of three main JAR files in runtime: `iesservr.jar`, `iesclien.jar`, `iescommn.jar`.

These JAR files contain Java methods that are the base foundation for the Scripting Engine's functionality, executed by the JVM. They are installed on the applications tier (on the Apache Web server) during the Rapid Install. For 3-tier JServer Technology Stack implementations, some manual post-installation implementation steps are required to be performed with these files.

For creating custom Java on a development workstation, some Java IDE tools may also require some of the Scripting foundation JAR files to be downloaded from the applications tier to the development workstation to add as libraries to the IDE. See JAR Files and the Java IDE above.

Three-Tier JServer Architecture

Since these JAR files provide the foundation for the Scripting Engine, these need to be deployed to the JServer in the database to provide the functionality of these Java methods to the JVM on the server in support of 3-tier JServer Technology Stack environments. Additional steps are required to grant explicit permission to identified agents to execute these methods. (The same permissions are required to be granted on any custom Java code so that agents can execute that code as well.)

These manual post-implementation steps (steps 6, 10 and 11) are described in the section Implementing Oracle Scripting > Implementation Checklists> Three-Tier Java Technology Stack Implementation Checklist.

Starting a Script in the Scripting Engine

Although not built with Oracle Forms, Scripting is accessed at the client tier by an agent using a Web browser to log into Oracle Applications, which relies upon the Oracle Forms technology stack. The agent invokes Scripting either through logging into an integrated Oracle Forms-based application, or through the Demo Form created for this purpose. Selecting a valid script then launches the Scripting Engine as a Java bean that executes in a stand-alone window.

The Scripting Engine, essentially, is the execution of the business logic of a script, which occurs in the Java Virtual Machine. In the Caching Architecture, the business logic is executed on the client tier (using the JVM provided by JInitiator); in 3-tier JServer Technology Stack environments, the business logic is executed using the JVM of the JServer. In both cases, the presentation logic executes on the client tier, displaying on the agent desktop in a Java-based window or Oracle Form, as discussed in the previous paragraph.

How Many Logins Are Required to Run a Script?

Scripting Engine users running Oracle Scripting 11*i* require logins to Oracle Applications. This login is associated with one or more ***responsibilities***, based on the functions the agent is to perform within the interaction center.

Oracle Scripting can be launched from integrated Oracle CRM business applications (such as Oracle TeleSales or Oracle TeleService). In such cases, these users would be required to be set up by the SysAdmin with the appropriate responsibility or responsibilities to access those Oracle Applications.

Oracle Scripting can also be launched from the Demo Form provided by any version of the Rapid Install for all Oracle 11*i* Applications. In order to see this Demo Form, an agent must be set up by the SysAdmin with a responsibility of `Scripting User`.

Beyond this information, a discussion of Oracle Applications responsibilities is beyond this document. Refer to other applications in the Oracle online library.

Agent Logins and the Caching Architecture

For implementations relying on the Caching Architecture, the login to Oracle Applications is the only login required.

Agent Logins and the Three-Tier JServer Technology Stack Architecture

For implementations relying on the 3-tier JServer Technology Stack architecture, a second login is required. This is true regardless of whether a script is initiated from an integrated application or from the Demo Form.

Upon selecting a valid script to launch, a Forms-based login screen will appear. Refer to this second login as the Scripting login. This login screen is similar to the Oracle 11*i* Applications login screen, with the same montage in the background and fields for a username and a password. However, the login information is (typically) *not* the same as the Oracle Applications login.

For 3-tier JServer Technology Stack operations, the agent must enter this information in order to instantiate the script. The username and password for the Scripting login are defined by a DBA (by running the `ies_create_agent_user.sql` script) as part of the Post-Installation steps for Agent Creation. Unlike the typical `apps` login, this username and password cannot be changed by an agent.

Use of the Scripting login is only required once per applications session for users of Oracle Applications 11.5.2 environments. Earlier environments will require a Scripting login for each Scripting session—that is, they will need to enter a username and password for database authentication each time a script is selected and the Scripting Engine launches. For those enterprises that must avoid multiple logins per session, upgrade to 11.5.2 (or higher) is recommended.

This will be obviated by the recommended migration to the Caching Architecture.

How Does a Java Application Run in an Oracle Form?

During the Rapid Install process, three Scripting-specific JAR files required for Oracle Scripting 11*i* to function, `iesclien.jar`, `iesservr.jar`, and `iescommn.jar`, are installed on the web server.

A "Wrapper" (included as one of the Java classes in the `iesclien.jar` file) allows Scripting to be embedded in an Oracle Form by wrapping the scripting GUI in Java code. This serves to expose the script to the Oracle Form using the Bean area.

From a Forms perspective, the Bean Area that invokes Scripting must be configured to specify the Wrapper class as its Java Wrapper. Then, when the Bean Area is initialized, it instantiates the Wrapper class. Subsequently, all communication between the Bean Area and the Java code occurs through this Wrapper.

The first time scripting code is executed, the JAR files required for the client, `iesclien.jar` (including the Wrapper), `iescommn.jar`, and (for Caching Architecture operations) `iesservr.jar`, are cached on the client machine (in a path such as `<DRIVE>:\Program Files\Oracle\JInitiator[`*`versionN`*`]`).

Note that this may take several minutes, although subsequent launches should be nearly immediate.

Responsibilities

Every application has one or more *responsibilities*; for example, for Call Centers using the Oracle Customer Support module of Oracle TeleService, each agent would have a `Customer Support` responsibility set up for him or herself. Each individual launching Scripting from the Demo Form (described immediately below) must have a responsibility of `Scripting User`. The systems administrator for Oracle CRM Applications 11*i* must associate the `Scripting User` responsibility (included with the "Rapid Install") with a particular applications login.

Launching a Script in the Demo Form

The "Rapid Install" includes an Oracle Form in which a script deployed to the database can be launched. This Demo Form is typically used for the purposes of testing that scripts are deployed appropriately or for debugging scripts in development. Theoretically, the Demo Form can also be used by interaction center agents to launch scripts that do not integrate with other Oracle applications. Running the Scripting Engine in stand-alone mode is not recommended, however, as the benefits of contact management, computer-telephony integration (CTI), and other features of existing Oracle applications that integrate with scripting are not available directly to Scripting and would be a duplication of complex functionality that already exists in the Call Center Applications suite. For more information on Scripting integration, see Scripting and Integration of Other CRM Applications in this document.

For step-by-step procedures for launching a script, see How to Launch Oracle Scripting below.

Launching a Script in an Oracle CRM 11*i* Application

If using an Oracle CRM application integrated with Oracle Scripting, launch that application using the appropriate apps login, and then start scripting in the manner prescribed by that application. See the appropriate Concepts and Procedures manual. For step-by-step procedures for launching a script, see How to Launch Oracle Scripting below.

How to Launch Oracle Scripting

1. Log into Oracle CRM 11*i* applications using a login that has been given the appropriate responsibility.
   - **For Scripting Demo Form users,** select the `Scripting User` responsibility.
   - **For users of Scripting-integrated business applications** (for example, TeleSales), select the appropriate responsibility to launch that application (e.g., `TeleSales Agent`).
2. Select the appropriate responsibility from the list of applications.
   - **For Scripting Demo Form users:**
     - The Demo Form will open.
     - Disregard any Script Information or Database Information fields that may display in the Demo Form.
     - Click the Script Chooser button, which will result in the Scripting Window (a list of available scripts).
   - **For users of Scripting-integrated business applications**:
     - In the application that opens as a result of selecting the appropriate responsibility, launch Scripting in the manner prescribed.
     - The Scripting Window (a list of available scripts) will open.
3. From the Scripting Window, select the appropriate script and script language combination from the dropdown field and click the Start Scripting button.
4. ***This step applies to 3-tier JServer Technology Stack implementations only.*** A Forms-based Oracle Applications login window will appear. Log in with the Scripting agent login set up by the systems administrator or DBA. For more information on this second login, see How Many Logins Are Required to Run a Script?
5. Based on the profile settings for your environment, the script will launch. You should be aware of a configurable variable, the panel display mode. This parameter controls whether the script displays multiple panels (with the active panel having focus) or displaying only the active panel.
6. Run through the script to ensure the Scripting Engine is functioning properly.

Removing Old Scripts from the Database

Old scripts and accompanying data can be removed from the Scripting schema in the database by executing a SQL script with that request. For this task it is recommended that you execute a script developed by a qualified DBA fully knowledgeable about the particular environment in question. You will need to be familiar with the Scripting schema to determine what portions of the script may be appropriate for your particular environment. A script is provided below *as a sample only*, that you may choose to use as the basis for constructing your script.

Note: Oracle Corporation provides this script as a *sample only*, from which you can build an appropriate script for your specific environment. Oracle Corporation is *not responsible for any damage or loss of data* that may occur from execution of this script or your customized version. Use at your own risk and with the consultation of a qualified DBA familiar with your environment.

Sample SQL Script for Removing Scripts and Data from Database

```
select dscript_id, dscript_name, active_status from ies_
deployed_scripts;

accept scriptId PROMPT "Please enter dscript_id for the
script that needs to be deleted: "

delete from ies_question_data where transaction_id in
(select transaction_id from ies_transactions where dscript_
id = &&scriptId);

delete from ies_panel_data where transaction_id in (select
transaction_id from ies_transactions where dscript_id =
&&scriptId);

delete from ies_questions where panel_id in (select panel_
id from ies_panels where dscript_id = &&scriptId);

delete from ies_answers where lookup_id in (select lookup_
id from ies_lookups where dscript_id = &&scriptId);
```

```
delete from ies_transactions where dscript_id = &&scriptId;
delete from ies_lookups where dscript_id = &&scriptId;
delete from ies_panels where dscript_id = &&scriptId;
delete from ies_deployed_scripts where dscript_id =
&&scriptId;
commit;
```

Integrating Oracle Scripting

Since Oracle Scripting is a 100% Java application, it can be integrated with other Oracle CRM 11*i* products. Oracle Scripting is sold as part of the Call Center Applications suite and is not recommended to be used as a stand-alone product.

Oracle Scripting provides an extensive set of APIs and events for external applications to monitor and control the flow of the script.

Scripting interacts with Forms 6.0 applications via the *Bean Area* (in Oracle Forms 4.5, this was called the *Java User Area).* The Forms-based application includes a Java wrapper that runs in the Bean Area and manages the communications between Scripting and the Forms-based application. This Java wrapper allows the application to call Scripting APIs and allows Scripting to pass data from the script to the application.

Scripting and Integration of Other CRM Applications

When we say that Scripting is "integrated with other CRM applications" we mean that any of the following activities can be performed:

- The application can call a script
- A Forms-based application can display one or more elements of the Scripting UI component beans (e.g., Panel Display bean and Progress Area bean; for others, see the discussion on *Script GUI components factor*y in the Scripting API document referenced below) within a tabbed form within the CRM application (or a separate Scripting window). Thus, from a user point of view, it looks like the Script is actually another Form within the application
- The script can share data (via Data Sources) with the CRM 11*i* application
- The script can launch (or "pop") a specific form in the CRM 11*i* application based on the call context

- The application can pop a specific Group within a Script (e.g., to a certain predefined set of panels or other Scripting functions)
- The application can initiate events within the script (such as start and stop a script), using the published APIs

The degree of integration varies among the various CRM applications. Check the documentation for each appropriate application to determine to what degree they are integrated with Scripting.

Current Integrations Between Scripting and Other CRM Applications

For Oracle CRM release 11.5.2 (also known as R2), Oracle Scripting is integrated with two applications in the Oracle eBusiness Suite: the Oracle Customer Support module of Oracle TeleService, and Oracle TeleSales.

.

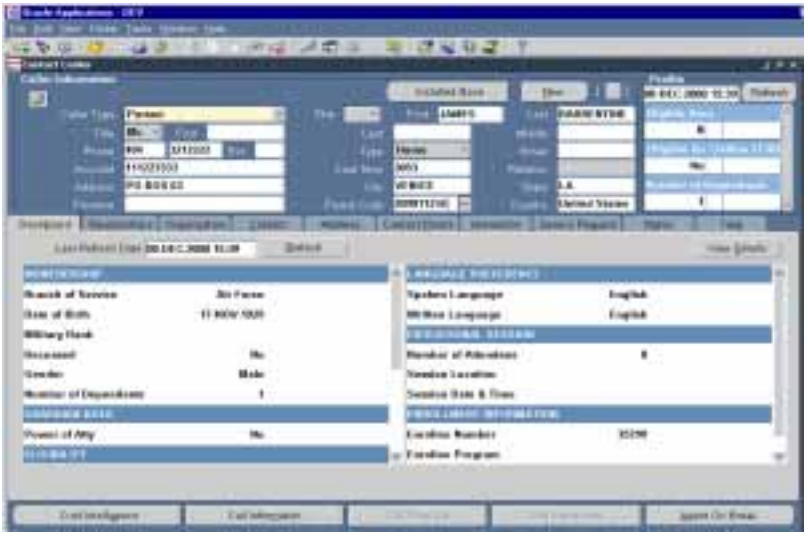


Figure 7. Customer Support module of Oracle TeleService.

Integration with Oracle TeleService has been increased since its debut in 11.5.1. Initially there was a single script designed to create a service request for Customer Support. When the script button on the tool bar was clicked, the service request script would launch, resulting in a service request being generated. Environments

upgraded to 11.5.1 with the Rolled-Up Patch (RUP)-B, and new 11.5.2 environments, experience a greater degree of integration; when the script button on the tool bar is clicked, you are now presented with a drop-down choice of scripts available to TeleService.

This Scripting Window may also be accessed by selecting Script from the Tools menu. Oracle TeleSales allows users to invoke scripts by selecting either the View Script button or by selecting a script from the menu that results from clicking the All Scripts button in the Overview tab, as shown in Figure 8 below. Alternatively, a script can be set up to launch automatically when the campaign it is associated with

Figure 8. Overview tab displayed in TeleSales application, from where a script can be launched with either of the buttons displayed.

How Oracle Forms Applications Pass Data To Scripting

TeleSales Example

A primary method for passing information from an Oracle Forms-based business application to Scripting is through an Oracle Forms command. For example, the following Forms command is executed from Oracle TeleSales whenever a script is instantiated (e.g., whenever a script is launched from TeleSales as mentioned above):

```
load_single_parameter('PARTY_ID');
load_single_parameter('CONTACT_ID');
load_single_parameter('PARTY_CONTACT_ID');
load_single_parameter('CUST_ACCOUNT_ID');
load_single_parameter('ADDRESS_ID');
load_single_parameter('CAMPAIGN_ID');
load_single_parameter('CAMPAIGN_SOURCE_CODE');
load_single_parameter('CAMPAIGN_OFFER_ID');
load_single_parameter('CAMP_OFFER_CODE');
load_single_parameter('ORDER_ID');
load_single_parameter('QUOTE_ID');
load_single_parameter('OPP_ID');
load_single_parameter('LEAD_ID');
load_single_parameter('COLLATERAL_ID');
load_single_parameter('EVENT_ID');
load_single_parameter('EVENT_REG_ID');
load_single_parameter('AST_RESOURCE_ID','GLOBAL');
load_single_parameter('AST_SALESREP_ID','GLOBAL');
     END ;
```

The Scripting session is invoked by using StartSession public Scripting APIs, and the values unique to the selected customer in TeleSales are sent to the Scripting Blackboard immediately thereafter using public Blackboard APIs. These APIs are published in the External APIs and Events Design Specification for Oracle Scripting document. For more information on how scripting functions with integrated products, refer to the specific documentation for that product.

Current Integration With Non-Forms Applications

Oracle iSupport is also integrated with Scripting to some degree. Note, however, that this integration is an entirely different matter from an architectural standpoint, since iSupport is not a Forms-based application. Essentially, iSupport provides a

web GUI powered by the Scripting Engine. For more information, refer to iSupport documentation.

Future Integration

Future releases of Oracle CRM Forms-based applications such as Oracle Banking Center and Oracle Collections Center are expected to include additional Scripting integration by release 11.5.5. On the horizon are additional integration efforts—more “canned“ scripts for those products that provide them, and brand new integrations. Oracle Scripting has a published set of Scripting APIs that allow other applications to share information with Scripting. Some products in the eBusiness Suite must still be modified to function with Scripting. Please refer to the most recent documentation for each individual product listed above regarding integration methods and the degree of integration with Scripting.

Scripting and Customization

Typically, a customer of scripting has included the customization of at least a single script to be developed by Oracle Consulting Services (OCS). Due to the nature of Scripting, these scripts are not considered to be supported by Oracle Support Services (OSS) unless (a) the issue is related to the actual product and not the customized script, or (b) such support has been arranged within the scope of a contract.

Modifying ˜Canned˜ Scripts

Even if Scripting is integrated out-of-the-box with a specific application, customizations to those scripts included with the application (e.g., during an OCS engagement) are not considered to be supported unless such support is specifically contracted for. Modifying “canned“ scripts for a given application is not without its own inherent risks. Each Oracle product development team understands the APIs, events, and data passed by its application and Scripting, and scripts to be provided with those products have been thoroughly tested. Consulting teams are not always privy to the same expertise, familiarity, and technical resources available to development teams of their respective products, and complications may effect scheduled delivery at great economic cost and with other contractual risks.

Related Products and Components

The following is a discussion regarding related products and components that affect the implementation and ongoing operation of Oracle Scripting.

Internet Web Browser Requirements for Oracle Applications

As of this writing, Oracle Applications 11.5.3-compliant Web browsers include Netscape Navigator 4.6 (or higher) or Internet Explorer 5.0 (or higher). The choice of Web browser must rely on overall interaction center requirements, and requirements for use of other applications with browser-specific dependencies. As always, the latest certified product lists, patches, and release information can be found on Oracle MetaLink at http://metalink.oracle.com.

Oracle JInitiator

When you access certain parts of the system (for example, those based on Forms or Discoverer), you will need Oracle JInitiator to supply the Java Virtual Machine. Since Oracle Scripting typically runs in either a Demo Form (for testing) or in consonance with a Forms-based business application, using Oracle JInitiator is a requirement of Oracle Scripting.

What Does JInitiator Do?

Implemented as a plug-in (Netscape Communicator) or ActiveX component (Microsoft Internet Explorer), Oracle JInitiator allows you to specify the use of the Oracle Java Virtual Machine (JVM) on Web clients instead of having to use the default JVM of the browser.

When Oracle JInitiator is needed, the browser attempts to load it. This will happen the first time a user accesses the Oracle Applications Web page to launch the Forms

applet; if Oracle JInitiator has not been previously installed, the browser downloads the necessary installation executable to the PC. Once installed, Oracle JInitiator runs the Oracle Forms Java applet and starts an Oracle Applications session.

Which JInitiator Version Should You Use?

Note: *Information regarding product dependencies and certification are likely to change without notice.* The latest information about what products or components are certified can always be found on MetaLink. The information below has been provided in an attempt to convey late-breaking information as of the publication of this document. However, it is your responsibility to refer to MetaLink for updated information.

At the time of this writing, three versions of Oracle JInitiator have been certified for use with Oracle Scripting: 1.1.7.27, 1.1.7.32, and 1.1.8.3. There are interdependencies regarding which version of JInitiator, which version of Scripting, and which architecture is being implemented. The matrix below is specific to using JInitiator with Oracle Scripting.

Oracle JInitiator Certification Matrix for Oracle Scripting

JInitiator Version	Scripting Architecture	Windows OS Supported	Functions With Oracle Scripting Versions:
1.1.7.27	3-tier JServer Architectures	95, 98, NT	11.5.1, 11.5.2, or 11.5.3 and up.
1.1.7.27	Caching Architecture	95, 98, NT	11.5.3 plus Minipack E, and up.
1.1.7.32	3-tier JServer Architectures	95, 98, NT	11.5.1, 11.5.2, or 11.5.3 and up.
1.1.7.32	Caching Architecture	95, 98, NT	11.5.3 plus Minipack E, and up.
1.1.8.3	Caching Architecture ***and*** 3-tier JServer Architecture	95, 98, NT, 2000	11.5.3 plus Minipack E, and up.

General Note 1: Applying Minipack E (patch 1563320) upgrades Oracle Scripting 11.5.1 and 11.5.2 to an "11.5.3 plus patch" level.

General Note 2: "11.5.3 and up" includes Oracle Scripting 11.5.4.

General Note 3: The Caching Architecture works with all three certified versions of JInitiator listed above. However, you must use Oracle Scripting "11.5.3 plus patch" (you must upgrade to Minipack E) in order to use the Caching Architecture.

General Note 4: Running Oracle Scripting with JInitiator 1.1.8.3 requires installation of Minipack E.

JInitiator 1.1.7.27 Notes: Certified for 3-Tier JServer Tech Stack environments with Oracle Scripting 11.5.1, 11.5.2, 11.5.3, and up. Certified for use with Caching Architecture with Oracle Scripting "11.5.3 plus patch," and up. Enterprises using 3-Tier JServer Technology Stack architecture may encounter the following:

ISSUE 1: If one agent logs into Oracle Applications and downloads JInitiator 1.1.7.27, and subsequently a second user logs into an NT session on the same NT workstation and attempts to log into Oracle Applications and then to Oracle Scripting, these users may encounter out of memory Java exceptions and should upgrade to JInitiator 1.1.7.32 after coordinating with OSS.

ISSUE 2: Users of Microsoft Internet Explorer in Cached Architecture environments may encounter an issue with server-based JAR files that have *not changed* downloading unnecessarily when Oracle Applications is launched. Upgrading to JInitiator 1.1.7.3.2 can resolve this issue.

JInitiator 1.1.7.32 Notes: Certified for 3-Tier JServer Tech Stack environments with Oracle Scripting 11.5.1, 11.5.2, 11.5.3, and up. Certified for use with Caching Architecture with Oracle Scripting "11.5.3 plus patch," and up.

JInitiator 1.1.8.3 Notes: Certified for Caching Architecture *and* for 3-Tier JServer Tech Stack environments only with Oracle Scripting "11.5.3 plus patch" and up. See General Note 1 above.

ISSUE 1: Certified for either architecture *only when updated with MiniPack E.*

ISSUE 2: This version of JInitiator ***required*** for clients running Windows 2000.

Where Can You Find JInitiator?

The latest version of JInitiator that was available at the time of the product release is on the Oracle Applications Rapid Install Server Technology CD in the `/util/jinitiator` path in the `Common` directory of your file system.

For More Information Regarding JInitiator

For more information on the latest certified version of JInitiator, check with MetaLink or contact your OSS representative.

Steps to Enabling the Java Console

A Java console can be enabled from the JInitiator control panel that enables the user of the Scripting Engine to see JAR files caching on the client machine. Errors that might be encountered when launching the Scripting Engine (or other Oracle Applications that require JInitiator) can be detected in the Java console.

1. Exit all instances of the Web browser.
2. If you are using other browser-related utilities (such as Netscape Messenger mail client, or browser-based newsgroup readers), exit those as well.
3. Open the JInitiator Control Panel (Start > Programs > JInitiator Control Panel <Version Number>).
4. Click on the Basic tab.
5. Ensure that the Show Java Console checkbox is selected.
6. The Java console will appear the next time the browser is started and JInitiator is invoked.

With experience, users can diagnose messages in the Java console and take steps to correct any problems. At times, the Java console may not yield enough information to determine what difficulty is being encountered. In these instances, review the trace files on the database server. For more information, see Troubleshooting—Enabling Trace Files in this document.

Oracle JInitiator Cache

Before using Oracle Scripting (after it has been run once), you must clean the JInitiator cache. This should also be done whenever Scripting-required JAR files (`iesservr.jar` and `iescommn.jar`) are reloaded into JServer while granting those specified agents permission to execute the JAR files in question.

Possible JInitiator Out of Memory Errors

With the current version of JInitiator (1.1.7.27) it is possible for Scripting to stop working when the JInitiator JAR file cache runs out of space. To see this error, the Java console must be visible and will display “Out of Memory“ errors. If this occurs, follow the steps below to empty the JInitiator cache.

Steps to Emptying the JCache

1. Exit all instances of the Web browser.
2. If you are using other browser-related utilities (such as Netscape Messenger mail client, or browser-based newsgroup readers), exit those as well.

3. From the filing system of the agent workstation, navigate to the directory in which JInitiator is installed. While the path for your configuration may differ, this directory path is typically similar to: `<DRIVE>:/Program Files/Oracle/Jinitiator<version>/`
4. Manually select and delete the entire contents of the `jcache` directory within the JInitiator parent directory.

Once Oracle Applications are run again, the JAR files will again cache in this directory. This may be a slow process, but once the JAR files are cached, each subsequent access of the JAR files will be from the local cache and will be much faster.

How to Configure JInitiator to Avoid Out of Memory Errors

To avoid this problem, it is recommended to set JInitiator heap and cache size parameters to recommended settings in the JInitiator Control Panel. Heap size is the pool from which memory is dynamically allocated. The cache is where JAR files downloaded from the server are stored.

1. Exit all instances of the Web browser.
2. If you are using other browser-related utilities (such as Netscape Messenger mail client, or browser-based newsgroup readers), exit those as well.
3. Open the JInitiator Control Panel (Start > Programs > JInitiator Control Panel <Version Number>).
4. Click on the Basic tab.
5. In the text field labelled Java Run Time Parameters, look for an option that begins with `-mx`. If it exists, make sure it reads **`-mx128M`**. If there is no **`-mx`** parameter in this text field, then add a new option to the end of the field to read **`-mx128M`**. With this action you are setting the minimum heap size to 128 megabytes.
6. In the same text field, Java Run Time Parameters, look for an option that begins with **`-Dcache.size=`**. If it exists, make sure the amount of cache allocated is 80 megabytes (80M). If it does not exist, then add a new option to the end of the field to read **`-Dcache.size=80M`**. With this action you are allocating 80 megabytes for JAR files to cache on the local machine.

Steps to Configure Proxy Server in JInitiator Control Panel

1. Open the JInitiator Control Panel (Start > Programs > JInitiator Control Panel <Version Number>).
2. Click on the “Proxies“ tab.

3. Ensure the “Use browser settings“ checkbox is disabled.
4. In the HTTP data fields, set the HTTP Proxy server hostname under “Proxy Address“ and the HTTP port number under “Port.“

Web Browser and Proxy Server Settings in the Scripting Engine

Web Browser in the Scripting Engine

The Scripting Engine includes an Internet browser licensed as a third-party product from ICEsoft. This ICE browser is accessible by clicking on the binoculars icon in the Scripting Engine tool bar. Upon this action, the browser will populate the bottom half of the Scripting Engine window. It can be resized to full screen or to dimensions desired by the user, and is dismissed by again clicking the browser icon.

This browser does not include a location field. However, the browser does include hypertext and hyperlink capabilities from its default home page (subject to the controls that may be established through any corporate firewall).

How to Configure Proxy Server Settings

If Scripting is installed at a site that is protected by a firewall, the ICE browser will not be able to access web pages outside the firewall unless its proxy server settings are configured properly. If extranet access is required for enterprises with a corporate firewall, you must manually configure settings by opening the JInitiator Control Panel for the version of JInitiator upon which your Scripting implementation is dependent. For instructions on how to accomplish this, see Related Products and Components > Oracle JInitiator > Steps to Configure Proxy Server in JInitiator Control Panel above.

Steps to Verify Proxy Server Settings

Once this is set, the user can verify that the settings are correct by looking in the Java console when loading jar files--each line will be followed by the proxy settings.

Accessing Scripting Through a Firewall

Oracle Scripting is designed to work inside a corporate firewall. It does not allow agents to run Scripting from a client machine outside a firewall due to Scripting’s use of the IIOP protocol for client to server communication. This affects any implementation in which an enterprise has an internal or external firewall that separates the client from the Oracle 8*i* database.

Can a Default ICE Browser Home Page Be Set?

At this time there is no way to set the default home page for the ICE browser. This capability is under review and we hope to incorporate this feature in a future release of Oracle Scripting.

Oracle RDMBS Requirements

Oracle Scripting Requires Oracle 8*i*

Oracle Scripting stores and retrieves data in the scripting schema created during a rapid installation, which includes the creation of an Oracle 8*i* database. Oracle 8*i* RDBMS is the database foundation of the CRM 11*i* suite of applications.

Integration With Pre-8.1.6 Oracle Databases

Note that data used for Scripting applications can be accessed in tables stored in versions of Oracle database earlier than 8.1.6. For example, data may be stored in custom tables in versions of Oracle RDBMS prior to 8.1.6, which are used by a legacy application and accessed by a script.

Oracle JDeveloper

Developing custom Java code to be used with Oracle Scripting requires access to a JDK 1.1.x or 1.2 compatible full Java Integrated Development Environment (IDE). The most popular Java IDE to be used with Scripting is Oracle JDeveloper. This product, licensed by Oracle from Borland and substantially improved, provides a full development environment, including editing, compiling, and particularly excellent debugging capabilities for developing and testing custom Java methods.

Note that for development of custom Java associated with Oracle Scripting, JDeveloper must be configured with two Scripting-specific JAR files. After importing these libraries and configuring JDeveloper to recognize them, the debugging utility will recognize methods included in these libraries. (Without importing these libraries, custom code calling the Scripting-unique methods will not compile.) The two libraries recommended for importing into JDeveloper for Scripting development are `iesclien.jar` and `iescommn.jar`. These files are installed on the applications tier of the server during the Rapid Install and must be manually copied (via FTP or any other appropriate method) to the Java development workstation. For information on how to add libraries to Oracle JDeveloper, refer to the appropriate JDeveloper documentation.

Best Practice Scripts

What Are Scripting Best Practices?

Best Practices Scripts are a library of Oracle Script Author scripts and associated Java and PL/SQL files. They are included with installation of Oracle Script Author (11.5.2 or later). The Oracle Universal Installer creates a `BestPractices` directory within the Script Author home directory. (Assuming the Script Author was installed in the Oracle Home directory, the directory path is: `[OraHome]\apps\ies\author\BestPractices`).

Best Practice Scripts are provided as *samples* indicating possible ways to develop different call flows that may be required in interaction centers. Developers of scripts may choose to reference the Script Author sample scripts and associated sample Java methods and PL/SQL commands provided with the Author tool.

To better reflect their nature as components you can use to help you build your own scripts, as opposed to being stand-alone scripts, Best Practice scripts will be referred to in future versions of Oracle Scripting as “Building Blocks.“

Can Best Practice Scripts Be Used as Installed?

Best Practice Scripts are intended to serve as sample building blocks for script development and are *not* intended to be used as delivered. They are provided as templates that knowledgeable developers can use as an aid to Rapid Application Development (RAD).

Note: Do not attempt to modify, deploy or use Best Practice Scripts until you have thoroughly tested your Scripting implementation as described in this Implementation Guide. Your environment must be confirmed to be fully operational.

What Is installed?

Fourteen scripts, eight Java files, two PL/SQL packages, and one README.TXT file are included in the `BestPractices` directory installed on the Windows NT, 95, 98, or 2000 Script Author workstation by the Oracle Universal Installer.

Who Should Use Best Practice Scripts?

The same skills required to create (or modify), deploy and execute scripts with custom Java (such as the Regression Test Script) and PL/SQL are required to use Best Practice Scripts. This includes:

- Knowledge of using and setting up a functional Java IDE such as Oracle JDeveloper
- Knowledge of creating syntactically correct Java methods
- Ability to reference packages and import statements
- Experience compiling Java code
- Familiarity with creating `.jar` files
- Ability to deploy compiled `.class` or `.jar` files to the Oracle 8*i* database.
- Experience with SQL commands and querying, inserting, or deleting records from tables or views in an Oracle database
- Knowledge of PL/SQL and how to package and reference PL/SQL commands in the Oracle 8*i* database.

How to Use Best Practice Scripts

Note: Best Practice Scripts are *not supported by Oracle Support Services.* The inclusion of these scripts with the product is intended to provide familiarity with possible approaches to execute functionality that may be desirable to implement within an interaction center. Some scripts may require parameters, database values, and so on as prescribed by the custom methods or database calls in the script. Best Practice Scripts are provided as-is; Oracle Corporation does not guarantee the scripts will function in every environment.

In the Script. Open any Best Practice Script in the Script Author to determine its function. Most likely you will need to view Java commands or PL/SQL commands referenced in the Script Author document to determine exactly what information is being passed to or from the script.

In the Code. Each Java command will include a descriptive name of the Java method in the Name property and the class name and specific Java method name in the Command properly.

The Object Dictionary of any PL/SQL block will include the names of tables being referenced and include join conditions, database field constraints, and so on.

Viewing, Modifying, or Executing Java Included in Best Practice Scripts

To view or modify Java code, open the associated Java file in your choice of Java development tools.

In order to execute Java methods, script developers will be required to compile the Java files, creating `.class` (or `.jar` files that include compiled `.class` files).

To execute the Best Practice Scripts using Java methods, the location to which you must deploy the compiled code (the appropriate `.class` or `.jar` file) differs based on whether you are executing Scripting in two tiers or three. For Caching Architecture operations, the code must be deployed to the APPL_TOP of the Applications server, and the `.class` or `.jar` files should be referenced, with appropriate paths, in the configuration file `appsweb.cfg`. For 3-tier JServer Technology Stack operations, the custom compiled code must be deployed to the Applications database.

Eight Java source files are included with the Best Practice scripts. The scripts reference custom Java methods from throughout these various Java files. When compiled as currently written, some of these class files are required to be located in a particular directory structure (a package). The package can be changed or removed, as long as the reference in each Java command (from the Script Author perspective) correctly identifies the changed path of the class file.

Integration With ERP or CRM Applications or Legacy Databases

The PL/SQL code included with the Best Practice scripts references the Oracle 8*i* database schema as of Oracle Applications Release 11.5.2.

Some of the Best Practice Scripts contain data integration points to key database tables and views within the Oracle 8*i* database schema (as of Oracle Applications Release 11.5.2). If an enterprise is already using Oracle ERP or CRM applications that have been implemented using a Rapid Install (11.5.2 or updated to 11.5.2), these scripts can serve as samples to querying and updating the 8*i* database.

For those enterprises who have their own external ERP applications, the Best Practice scripts and code can be analyzed and used as a basis for integration with legacy databases.

For More Information on Best Practices Scripts

Refer to the *Oracle Scripting Concepts and Procedures* document for more information on Best Practice Scripts.

Regression Test Script

A test script with custom Java methods has been created by Oracle Corporation for use in testing a Scripting implementation. This Regression Test Script contains examples of virtually all commands that one can execute within a script. It was developed by members of Oracle's development team in order to both test and demonstrate how commands work within the context of Oracle Scripting. It serves as an excellent means of testing a Scripting implementation. Note that if you are using the Regression Test Script for purposes of testing the implementation in the prescribed order of implementation steps, this process allows you to become familiar with deploying custom code to the JServer. This must be performed each time custom code is created or modified in support of a script.

The Regression Test Script is available through the following methods:

Downloading the Regression Test Script From MetaLink

Registered users of Oracle Scripting have access to Scripting-related release information, patches, bug notifications, and testing tools online at the MetaLink web site, at http://metalink.oracle.com. The Regression Test Script on MetaLink has a Document ID of 124522.1.

Obtaining the Regression Test Script From Oracle Support Services

Registered users of Oracle Scripting who cannot locate the Regression Test Script for any reason should contact their OSS representative.

After Downloading the Regression Test Script

Download the Regression Test Script from one of the sources indicated above and unzip the compressed file.

What Is Included in the Downloaded File?

Included in this compressed file are three files: the latest `.script` (Script Author) Regression Test Script file, a `.java` file that contains the custom methods to support the script, and the associated `.class` file which is the compiled version of the supporting Java code.

- Open the `.script` file in the Script Author and deploy this script to the applications database installed during the Rapid Install process. (Note that the script will be deployed to the database as—and called by—the name entered into the `Name` property within the `Script Properties` dialog accessible from the `File` menu.
- You need take no action with the included `.java` file. Typically, one will not deploy the Java file into the JServer. Keep Java files in the event you must modify your custom code; a compiled file cannot usually be modified.
- Load the `.class` file into the JServer using LoadJava. Note that it is perfectly acceptable to first compress this into a `.jar` file using the utilities provided for this purpose in the Java Development Kit (JDK).

NOTE: When deploying to the JServer custom code such as the compiled Java class file supporting the Regression Test Script, you must include explicit `-grant` privileges for agents that have been created for the enterprise. If you exclude this step, the Regression Test Script can only be run and tested with a Scripting login for the `apps` user.

Planning an Oracle Scripting Implementation

Purpose for Oracle Scripting Implementation Planning

The importance of implementation planning for Oracle Scripting cannot be overemphasized. The primary purpose for this planning is the successful implementation of Scripting. As noted in the Overview section of this document, Oracle Scripting differs from other CRM applications because it is not an application, but a development tool designed to create and execute call guides for customer interaction centers. Thus, the process of implementing scripting is actually a part of the customization process that Consulting must undergo as part of any new CRM implementation.

Implementing scripting is a complex process that requires knowledge of a variety of technologies and processes. Without the appropriate planning it is easy to mismatch requirements, underestimate the appropriate hardware, software, or network configuration, or end up with substandard system performance.

AIM's Role in Scripting Implementation Planning

Fortunately, Oracle customers and consultants have access to a powerful tool for success: the Application Implementation Method (AIM). The best way to ensure a successful implementation is to follow the AIM methodology. AIM helps consultants or an enterprise to collect, analyze, and adjust all aspects of an implementation: business rules and processes, financial and scheduling aspects, technical requirements, analysis and design, etc.

Oracle customers who are interested in licensing AIM or would like more information about this methodology should contact their sales representative. This document will make frequent reference to AIM documentation, using the two-letter

software life cycle designation, followed by a pointed serial number, e.g., “RD.050,“ a Requirements Definition document entitled “Gather Business Requirements.“

Mapping Implementation Planning to AIM

Scripting implementation planning requires careful consideration of the following:

- Assessing requirements, scheduling, resourcing (e.g., CR.010, CR.030)
- Business requirements mapping (e.g., BR.020, BR.030)
- Understanding the application architecture (e.g., MD.070, MD.080)
- Understanding the installation and configuration process (e.g., BR.100)
- Verifying and testing the implementation (TE.040, TE.090)

Scripting Implementation Project Scoping

Scripting-Specific Scope, Objectives, and Approach

At the beginning of planning an implementation, it is strongly recommended that Oracle Consulting develop, in coordination with the enterprise implementing Scripting, a Scope, Objectives, and Approach (SOA) document (AIM CR.010) *specifically for the Scripting portion*, to help delineate the scope of the scripting effort up front.

Scoping Considerations

Clear, concise communication of the scope of the system is key. What is included in the custom implementation of the script? What is not included? What can be considered follow-on consulting work for a successive project, and has this been agreed to between consulting and the implementing enterprise? Does the customer intend to maintain the script, build more from scratch, or retain Oracle Consulting or Oracle consulting partners for any required changes? Is customer training and knowledge transfer included in the cost, schedule, and in the mindshare of both client and consultants? Who is responsible for developing test scripts? Who is responsible for executing them?

Fortunately, AIM comes through as a reliable method through which to communicate with the client regarding expectations.

Discovery Process

The actual scope of a Scripting project can be deceptive in nature. While at first it may seem that the scoping assessment effort may include just the business logic of the script and the particulars of the scripting implementation, a closer examination reveals that there are three components: the **text** (the question / answer portion of the script provided by the client for the call guide), the **logic** (hopefully worked out in detailed flow charts to ensure call and data flow), and the **technology layer**.

.

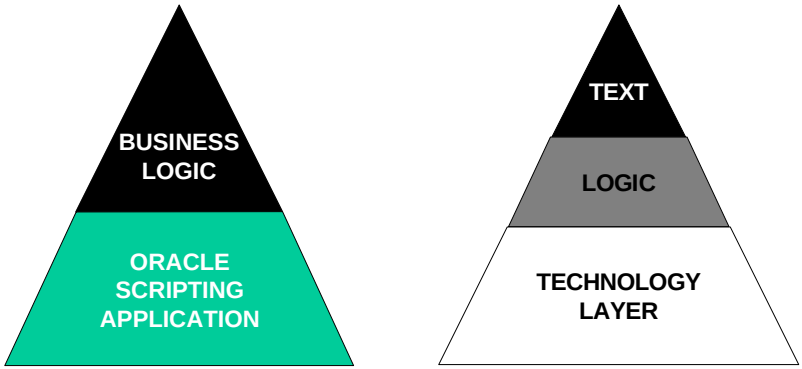


Figure 9. Early views of the Scripting scoping and discovery process

Text

The text of a script refers to the words to be spoken by the interaction center agent to the customer or prospect. In a call center this is often referred to as a "call guide." Customers may have strict requirements regarding the text. For example, a call guide may have been developed by a customer and approved by their legal department. This can present challenges to any emerging requirements to change the text or flow of a script. Customers and implementation project management should be aware that some changes are to be expected in the course of any implementation. Further, a substantial amount of changes to planned text may be required, for example:

- If the Scripting tool's functionality is not well understood. Particularly important to planning a successful script is an understanding of Scripting's question and answer paradigm.

- If the script has not been carefully analyzed using flowcharting analysis. Particularly important is the full accounting of every decision and its ramifications on the subsequent flow of the script. See the next paragraph.

Logic

The business rules must be thoroughly understood by those creating the text for a script so that all possibilities can be taken into account by those building the script. **This is best accomplished by a complete flowcharting of the intended script.**

In addition to flowcharting, and adding to the text component mentioned above, development of the logic may include specific navigating or unscripted instructions to an agent.

- Navigating instructions assist the agent in understanding where to navigate the script. For example, "After collecting all information and entering in the series of checkboxes below, click Continue."
- Unscripted instructions direct the agent to make a judgment in order to provide an answer in the current panel. They might also contain navigating instructions, such as in the following example: "Determine if this customer is eligible using the following criteria: (a) Of legal age, (b) Willing to sign the release form. If both are true, select Eligible button. Otherwise select Ineligible button."

Typically there is more freedom to create or revise instructions to agents in the course of a project than to change text. It is here that one can get around tight controls (such as approval to all text changes by an enterprise's legal department or review board) if they exist.

Technology Layer

The technology layer includes an analysis (from a Script Author perspective) of what methods are required to accomplish the branching and flow control of the script based on the logic. This includes an analysis of which and how many Script Author elements are required for a given script—panels, groups, blocks, and appropriate branching—but should extend beyond the tools available within the Script Author GUI. For example, one can attach commands to Scripting elements, but then typically code is required (PL/SQL, Java, Forms, etc.) to support those commands to implement the intended functionality of the script.

Business Analysis

Additionally, business analysis will be required throughout the project, often interrupting development progress. This is because a more detailed understanding of the system being constructed—only available mid-stream—is likely to result in

adjustments to the development approach, scheduling, and project resources. An experienced project manager will recognize these realities and plan for adjustment, rather than struggle with carrying out a plan conceived before a thorough understanding of the complexities of a scripting implementation were obtained.

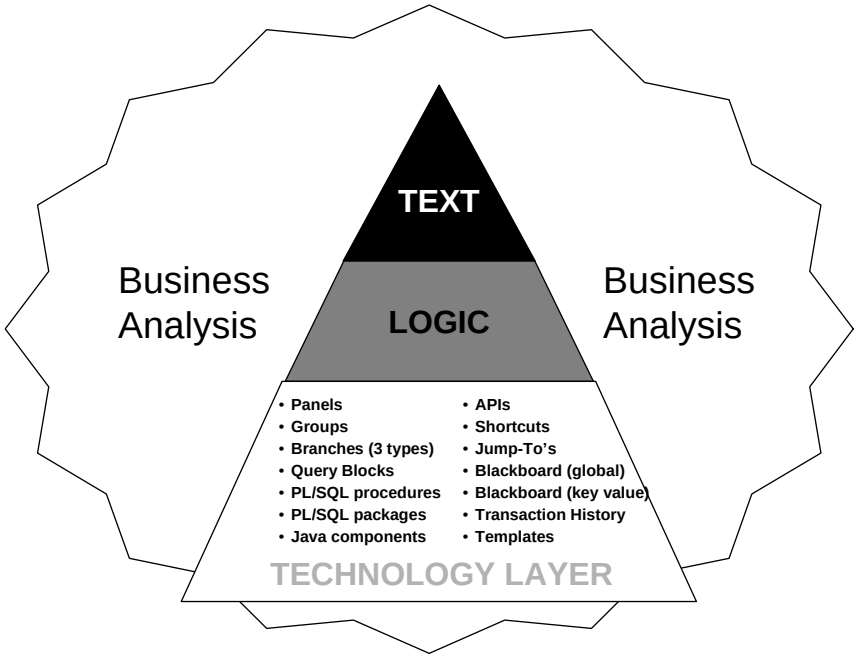


Figure 10. An informed view of Scripting scoping and discovery considerations

Key Elements of Success

It is the technology layer and the requisite business analysis that can require the bulk of time on a project, although confirming the text and logic *prior to building the script* are absolutely critical prerequisites. Also crucial to success from a project and resource management (considering timeliness and the availability of the appropriately trained project members) is the implementation of strict change control for the project once the scope has been agreed to.

The analysis of the technology layer reveals many technical aspects to be considered; it may be hard to scope them, and if the project allows, a multiple-phase process is recommend, which allows for benchmarking which can be used to determine average completion time for the appropriate tasks.

Integration Benchmarking

Oracle Scripting can be integrated with other Oracle CRM 11*i* applications in one of three ways:

- You can display data from another application within the context of the script;
- You can launch external applications from the script; or
- You can launch the script from other applications.

Ensure that you understand the nature of the desired integration. This includes a determination of whether the application exposes APIs that can be accessed by Scripting, and that is compatible with the technology stack.

If substantial integration requirements are indicated, it may also be in the best interests of the project to benchmark a limited integration to determine the true level of effort before scoping a larger integration effort. This will help in scheduling and resourcing.

Scripting Implementation Testing

A Scripting implementation needs to be tested not only for success of the implementation / installation, but also from the perspective of the custom script and any integration required. Different resources may be required, and consequently different phases may be planned for. This may require drafting separate AIM documents per phase.

AIM and Implementation Testing

The importance of thoroughly testing of the script cannot be understated. As part of any consulting engagement, time and resources should have been allocated for developing a solid testing strategy, and performing thorough system and integration testing. The business testing process is covered as part of the TE series of documents in the AIM methodology. The relevant AIM documents include:

- TE.010—Testing Strategy
- TE.020—Unit Test
- TE.030—Link Test
- TE.040—System Test
- TE.050—Systems Integration Test

Identifying Steps to Test Implementation

At the beginning of planning, it is strongly recommended that Consulting develop a Scope, Objectives, and Approach (SOA) document (AIM CR.010) specifically for the Scripting portion, to help delineate the scope of the scripting effort up front.

This will lay out a specific plan when it comes to the testing portion. At minimum, two AIM test documents should be developed: a testing strategy (TE.010) and a system test (TE.040).

Individual test cases should be planned for to test the functionality of the implementation (e.g., that the product is installed and functioning properly) as distinct from those test cases that examine the functionality of the customized script.

From a planning perspective, if substantial integration requirements are indicated, a TE.050 should also be required, and it may be in the best interests of the project to benchmark a limited integration to determine the true level of effort before scoping a larger integration effort.

At times it is negotiated between Oracle Consulting Services (or the Oracle consulting partner) and the implementing enterprise that the client shall be responsible for developing and performing system tests. Regardless of who develops and performs the test, both parties should carefully review the test plans (to determine that all aspects that should be tested will be tested) and the test plan results. If developed by the client, documents should be reviewed by subject matter experts from the consulting organization (with expertise in Oracle Scripting as well as any other CRM or custom application) to ensure success.

Where Does Testing Occur?

Oracle Scripting may not necessarily be exhaustively tested on site. For example, if the degree of customization and integration of components external to Scripting include only a few database calls, the system may be tested from the Oracle Consulting team's home base.

Implementation Testing in Context of This Document

At minimum, regardless of whether AIM testing strategy and system test plans are developed, it is incumbent upon those implementing Oracle Scripting to perform three tests. The first is to deploy and then execute the script developed as part of the exercise listed in the Creating a Script section of this document (or another simple script). This is to ensure that the Script Author has been installed correctly and the server is configured correctly to allow a script developer to deploy a script to and execute it from the database.

The second test is to deploy and execute a previously tested script with custom Java methods associated with it. The script must include calls to the custom Java methods (using Commands), and the associated compiled Java `.class` files (preferably compressed into `.jar` files) must be loaded into the JServer.

The third test is to deploy and execute the custom script, ensuring its functionality meets specifications (typically, as determined between AIM documents CR.010, TE.040, and perhaps MD.070 and MD.080).

Implementation Roles and Functions

There are four basic roles for implementing Oracle Scripting:

- Database Administrators (DBAs)
- Systems Administrators (SysAdmins)
- Script Authors
- Developers

Database Administrators

In order to properly implement Oracle Scripting, a DBA with knowledge of Oracle 8*i* Relational Database Management System (RDBMS) is essential. Specific required areas of knowledge include:

- Setting, logging and monitoring Trace Files;
- Understanding the Oracle Applications database schema;
- Making SQL calls to the database; and
- Writing and storing PL/SQL packages or stored procedures on the database.

Additionally, for implementing Scripting in the 3-tier JServer Technology Stack architecture, DBAs must have knowledge of:

- Setting up and implementing Java in the database (the JVM);
- Familiarity with the Multi-threaded server (MTS) features of 8*i*, particularly configuration of MTS shared servers and dispatchers. (For more information, refer to Note 69043.1 on MetaLink.)
- Establishing and administering the IIOP port;
- Familiarity with CORBA, IIOP, and JNDI; and

- Using LoadJava and DropJava utilities to move required code (such as JAR files containing Java class files which support specific Oracle Scripting scripts or call guides) into the database.

Systems Administrators

Setting up the scripting environment can be complicated, and essential to success of an implementation is the assistance of one or more SysAdmins, who must have knowledge or prior experience in:

- Applications environment preparation and setup;
- Applications mid-tier configurations;
- Understanding, configuring and maintaining the `appsweb.cfg` file;
- Configuring JInitiator Client settings; and
- Installing and patching Oracle Applications.

Script Authors

Script authors need to leverage the tools inherent in the Script Author development environment to build scripts that are often large and complex. Strong analytical skills, consistency, and the ability to develop scripts from complex flow charts are essential. Script authors should have a good understanding of solid programming concepts (including naming conventions and principals of design), although they should also have Developers and Systems Architects accessible to consult with regarding script development approaches and techniques. Constant assessment of requirements provided to the development team is also required of the script author. In instances where more than one person is developing a script, excellent communication skills are also required. See the section Challenges to Scripts Built in Parallel with Multiple Developers in this document for more details. The script author role is an excellent training ground, exposing staff to a variety of technologies that may soon enable persons in this role to expand her or his knowledge to the point where they can move into the more technical Developer role, as described below.

Developers

Developers are also required for most Scripting development and implementation projects. One main role of developers will be to perform integrations between Scripting and other applications (particularly Forms-based applications). These developers must be familiar with the Script Author development environment, as they may be called upon to develop complex scripts (or scripting groups within a

script) that less experienced script authors may not possess the technical skills to perform. Developers also may be required to embed complex logic in a script by developing custom code (Java, PL/SQL, database integration, etc.). This code would typically be associated with a script using the Commands dialog box in the Script Author. Such complex logic would be required to enforce business rules provided to the script development team, preferably in the form of carefully detailed flow charts. Constant assessment of the requirements provided to the development team is also required of the developer. These resources need to work in close collaboration with the chief script author(s) of the script, requiring strong communication skills in addition to technical demands.

End-User Roles and Functions

There are two basic end-users for Oracle Scripting:

- Call center agents, who use the runtime Scripting GUI to guide them through interactions with customers as they talk on the telephone
- Script authors, who actually develop scripts to meet the call center needs

Call Center Agents

Agents are the people who use the completed, runtime scripts as guides when they interact with their customers. Oracle Scripting presents a very simple GUI that makes it very easy for agents to navigate.

Script Authors

Oracle Consulting will typically deliver at least one “template“ or baseline script to the customer as part of the CRM implementation process. Regardless, most customer interaction centers have extremely urgent needs to rapidly create and modify scripts for their production environments.

Whether script authors are Oracle Consulting staff or employed by the licensed users of Scripting at the client site, the following areas of knowledge are strongly recommended for individuals developing scripts:

- Knowledge of solid programming concepts;
- Java development skills;
- PL/SQL and database programming;
- Integration with other Oracle applications and Oracle Forms; and

- Strong analytical skills and the ability to map out business flow for script planning.

Overview of Typical Business Processes for Oracle Scripting

The following business processes are involved in the use of Oracle Scripting:

- Script Creation
- Agent / Application Linkage
- Script Deployment

Figure 11. Business processes involved in using Oracle Scripting

Script Creation

Script creation refers to the process of defining the text, branching logic, and the technology layer for a script. The text and branching are a graphical representation of the question / answer flow through a *call guide* (a call center-specific term which refers to, in the case of Oracle Scripting, a script developed in Script Author to enable a call center agent navigate through a carefully-scripted set of interactions with the customer or prospect). The term *technology layer* refers to the additional considerations that must be made in defining a script for a consulting engagement. These tasks are associated with all of the logic and data manipulation that must occur in the background, transparent to the user. The development of the technology layer typically includes using one or more of the following types of procedures:

- Identify data needed from external applications—The script often needs to reference demographic, product, and other information needed during the context of the call. For instance, the customer's name and address may be validated as part of the script logic.

- PL/SQL processes and packages and Query blocks—To perform activities such as validating users, accessing data in external tables (when it's not already available via published APIs), including data in non-Oracle CRM applications.
- Java commands—To perform conditional logic, set up constants and other variables, and to manipulate both data and call flow as part of the customer/agent interaction.
- Blackboard commands—Scripting uses the blackboard, an area of memory within the Scripting Engine which stores values of embedded values, responses to previously-asked questions, and data passed to the script from external applications.
- Shortcuts—Quick access points, both within the script (for example, go to a wrap-up group) and external to the script (for example, jump to a specific URL on the corporate intranet).

To assist in some of the tasks associated with the technology layer, several template scripts have been defined, called Best Practices scripts. A library of Best Practices scripts is placed in the Script Author directory by the Oracle Universal Installer when installing the Script Author (11.5.2 or later). These define some commonly-used functions, including: customer demographic information, callback information, call wrap-up, and "do not call" templates. These templates can be reused or modified in new scripts, thus reducing the development tasks associated with these types of activities. However, any new script is likely to require additional data beyond that supplied in the Best Practice templates. For more information, see Best Practices Scripts in this document or refer to the Oracle Scripting Concepts and Procedures manual for more information on Best Practice Scripts and their locations.

Finally, the script must be integrated with the desktop application. Oracle Scripting is designed to act in conjunction with other CRM applications, and we do not recommend its use as a standalone product. The reason for this is that Scripting is not designed to act as a contact management system; it does not collect the detailed statistical data that is so crucial to most interaction center enterprises. It lacks detailed reporting, CTI statistics, data validation, and other benefits that are provided by CRM applications such as Oracle Customer Support (a module of TeleService), or Oracle TeleSales.

The processes associated with creating a script consist of the following high-level activities. Because they are typically performed by Oracle Consulting (or by an end user), standard requirements and design procedures should be followed to guide the creation process. See AIM's Role in Scripting Implementation Planning in this document for more information.

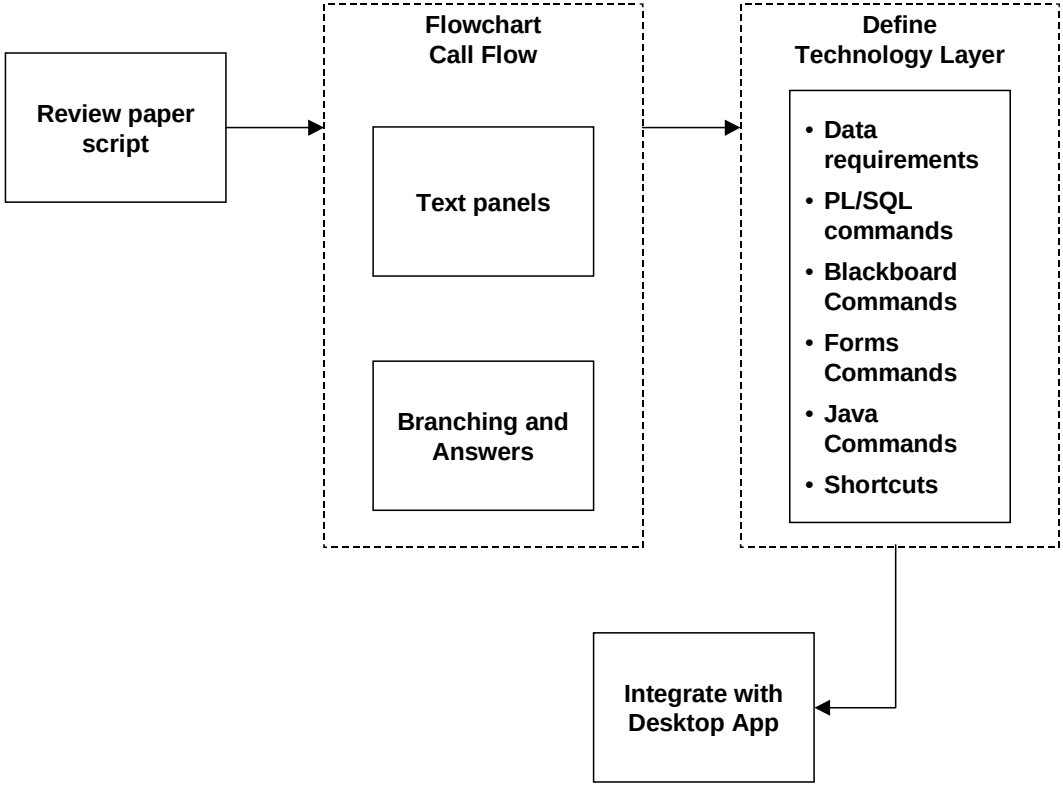


Figure 12. Processes involved in creating a script

Agent/Application Linkage

In order for an agent to use a script, the desktop application (Oracle TeleSales, Customer Support, etc.) must employ a method by which it determines which script to display to the agent. The script is most likely to be called based on one of the following:

- Agents are assigned to work on specific scripts and campaigns—more likely in an outbound telesales environment
- The application determines which script to run based on parameters in the customer database or on inbound DNIS or media type

Script Deployment

Scripts are deployed from the Script Author by a process that compiles the script and loads it into the Oracle 8*i* database, which also acts as the application server for the middle tier. This process is initiated from the Author tool by selecting **Deploy Script** from the **Tools** menu.

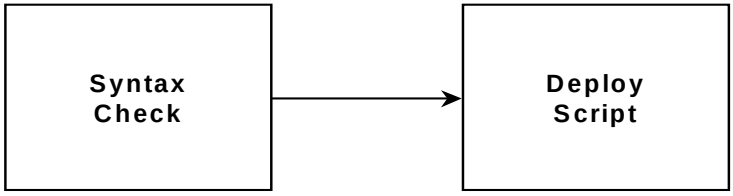


Figure 13. Selecting the Deploy Script command checks the syntax and, if successful, compiles the script and deploys it to the database

Implementing Oracle Scripting

This chapter is central to your ability to implement Oracle Scripting and is the heart of the implementation guide. First, this section contains information regarding the starting point for an Oracle Scripting implementation. This is followed by a description of how to perform the prerequisite step of setting system profile values (applicable to Scripting 11.5.2 and up). A detailed flowchart is included, to pictorially depict the steps involved in installing and implementing Scripting, and when to perform the relevant steps for the architecture required for your implementation. Then a checklist will help you tick off the procedural steps required to implement Oracle Scripting in the Caching Architecture. This is followed by a similar checklist for implementing Scripting in the 3-tier JServer Technology Stack architecture. *Note, however, that this approach is not currently recommended and will not be supported after April 1, 2001.* Both checklists are further supported by a detailed explanation of each checklist step. Finally, a checklist and the corresponding detailed steps for Converting Three-Tier Environments to a Caching Architecture are included, to aid enterprises already operating on the 3-tier JServer Technology Stack architecture to ensure conversion and continued support.

After you have met the starting point requirements, obtain required patches from MetaLink and refer to the flowchart for installation / implementation.

Note: Steps one through three listed in the Caching Architecture Implementation Checklist and detailed in the Detailed Caching Architecture Implementation Steps below are to be performed at the point in the flowchart designated “Caching Architecture Post-Install Implementation Steps.“

Steps one through twelve listed detailed in the Three-Tier Java Technology Stack Implementation Checklist below are to be performed at the point in the flowchart designated “3-Tier JServer Tech Stack Implementation Steps.“

The remaining steps in each checklist are to be performed (for testing or production) upon completion of any post-installation patching of other Oracle Applications 11*i* products that may be required. See MetaLink for the latest information or patches for each product.

Implementation Starting Point

At the start of any Oracle Scripting implementation, you must have installed:

1. The necessary Oracle 8*i* technology stack and Oracle Applications file system components on the server or servers.
   - You need to complete an administrative-tier installation on the primary applications server.
   - You need to complete a Web-tier installation on each server you will be using, including the primary server.

   This is accomplished by performing a Rapid Install.
2. The Oracle Applications products that are necessary for your Oracle Scripting solution on the server or servers.
   - The Script Author component of Oracle Scripting must have been installed using the Oracle Universal Installer (typically on a development workstation).
   - Oracle Scripting does not *require* any other applications in the Call Center suite. However, Scripting is typically used as part of a broader set of CRM applications in order to leverage the contact management, CTI, and full customer handling capabilities those products have to offer. These are typically installed during a Rapid Install, although each application may have its own manual post-install implementation steps, as indicated in the appropriate product documentation and release notes.

3. An Oracle Applications-compliant Web browser on the client (typically, the agent desktop, or a development workstation). For more information, see Internet Web Browser Requirements for Oracle Applications below.

4. Oracle JInitiator on the client.

   Technically, Oracle JInitiator is not prerequisite because the correct version for the appropriate environment may be downloaded when an agent logs into Oracle Applications at runtime. For more information, see About Oracle JInitiator.

Another prerequisite (for 11.5.2 and later implementations) is to establish system profile values, as discussed in Setting System Profiles below.

Additional Requirements for Caching Architecture Implementations

Caching Architecture implementations require a patch level for Oracle Scripting of Scripting Minipack E (patch 1563320). This and all other referenced patches can be found in MetaLink. If for any reason patches are not available on MetaLink, please contact your OSS representative. However, please always use MetaLink as your first recourse.

Note: Scripting Minipack E is a follow-on patch to Oracle Scripting 11.5.3. Applying this patch brings Oracle Scripting up to a level of Scripting "11.5.3 plus patch," regardless of which version of Scripting was installed. Earlier versions of Oracle Scripting cannot use the Caching Architecture. For scalability and support purposes, Oracle Corporation recommends upgrading all Scripting implementations to the Caching Architecture prior to April 1, 2001.

Additional Requirements for Three-Tier JServer Technology Stack Implementations

Prior to beginning implementation of Scripting, you must have available all information needed to implement for this particular architecture. Each of the following are requirements unique to 3-tier JServer Technology Stack implementations:

- List of all agents required to use Oracle Scripting (and their corresponding Oracle Applications logins and current responsibilities). This is required in order to generate a list of Scripting-specific usernames and passwords to be used to give each agent a database login (upon creating Scripting agents; see the Three-Tier Java Technology Stack Implementation Checklist, step 9).

Note that agents can typically change their Applications passwords unless this capability is deactivated by the Systems Administrator. The Scripting User password, unique to 3-tier JServer implementations, typically is set once in the step referenced above and is not changed.

- Resources knowledgeable in Oracle 8*i* database administration, including knowledge of MTS and of IIOP.
- Resources knowledgeable in enabling and administering Java in the database, including publication of the JNDI name and use of LoadJava and DropJava commands, with knowledge and privileges to execute LoadJava with `-grant`.

Setting System Profiles

For 11.5.2 or later implementations, ensure the Oracle Applications system profile values are appropriately established. When using 11.5.1, the parameters will need to be identified through other methods (for example, as values entered into the Demo Form).

1. Launch a compatible Web browser and log into Oracle Applications through the Personal Home Page (PHP) with the username and password for the system administrator (typically `sysadmin/sysadmin`).
2. Select My Home Page and scroll down through the Applications list of responsibilities. Click the `System Administrator` responsibility link.

> **Note:** This is not the same as the `System Administration` responsibility (emphasis added). If this responsibility is also listed, ensure you do not select it erroneously.

3. Upon clicking the `System Administrator` link, files will cache (the montage associated with Oracle Applications 11*i* will appear in the Web browser). When all files have cached, the Navigator screen appears, indicating the login responsibility in the title bar (for example, `Navigator - System Administrator`).
4. In the Navigator window, ensure the Functions tab is selected so that you can update system profile options.
5. Double-click Profile, and in the resulting child menu, select System. Then double-click System (or select the Open button).
6. The Find System Profile Values dialog is displayed, with only the Site check box selected. Leave this selected and also select the Application check box.

7. The Application field will highlight and an elipses (...) will appear. Select the ellipses (...) box to bring up a list of values.
8. Scroll down through the list and select Scripting; then click OK. The Find System Profile Values dialog is again displayed, with the Application field populated with a value of Scripting and with the Profiles with No Values check box selected.
9. In the Profile field, enter IES% and click the Find button.
10. The System Profile Values dialog box appears. The parameters indicated here are system profile settings for this particular environment. Different parameters are required for Caching Architecture or 3-tier JServer Technology Stack operations (as indicated below).

All seven profile settings displayed in the matrix below are designated as required fields by default. If the error "`Required profiles not set`" is returned, provide a value for the system profile in question, based on requirements for the implementation (including architecture type).

For example, even though a JNDI name is not referenced for Caching Architecture operations, this field is validated to ensure it is not null. You can use the 3-Tier JServer Architecture value indicated in the matrix below, or provide any other value. The Scripting Bean Display Mode parameter must always be set to display the Scripting bean in a separate form. The panel display parameter is also a required field (although which valid value is selected is not relevant from this perspective).

System Profile Parameters Matrix

Profile Value:	Site Value: Caching Architecture	Site Value: 3-Tier JServer Architecture
IES: IES ARCHITECTURE TYPE	`Two Tier Mode`	`Three Tier Mode`
IES: JNDI NAME USED FOR SCRIPTING APPLICATION	<Not evaluated; *use same value as 3-Tier*>	`/test/oracle/apps/ies/corba/common/Master`
IES: PORT FOR SCRIPTING APPLICATION	<TNS Port>	<IIOP Port>
IES: SID OF ORACLE SCRIPTING DATABASE	<Database SID>	<Database SID>
IES: SCRIPTING BEAN DISPLAY MODE	`Display Scripting bean in separate Frame`	`Display Scripting bean in separate Frame`
IES: SCRIPTING PANEL DISPLAY MODE	<To show one or many panels concurrently>	<To show one or many panels concurrently>

Profile Value:	Site Value: Caching Architecture	Site Value: 3-Tier JServer Architecture
IES: SERVER HOST NAME FOR SCRIPTING APPLICATION	<Database listener host>	<IIOP listener host>

Profile Value Notes

Further descriptions of the system profile options are included below. Note that in some new environments, three Oracle iSurvey settings (**IES: SVY ERROR PAGE**, **IES: SVY HEADER PAGE** and **IES: -ASSIGN UNSERVICED CPS**) may appear in this Form. Oracle iSurvey is a new product related to Scripting; iSurvey settings are not relevant to Oracle Scripting implementations, and these parameters should be left blank or with default values set.

IES: IES ARCHITECTURE TYPE: This indicates whether the JVM of the server (3-Tier) or the client (2-Tier) should be used. List of Values (LOV) options include `Two Tier Mode` and `Three Tier Mode`.

IES: JNDI NAME USED FOR SCRIPTING APPLICATION: The JNDI Name is only evaluated for 3-Tier JServer Technology Stack operations; however, a value is required for this system profile parameter by default. Include the (case-sensitive) JNDI name value indicated in the 3-Tier column.

IES: PORT FOR SCRIPTING APPLICATION: The port for Scripting application refers either to the TNS or IIOP port, and is also based on whether a 2-Tier or 3-Tier **IES ARCHITECTURE TYPE** setting is used. This topic is addressed in further detail in the Scripting and Port Numbers section in this document.

IES: SID OF ORACLE SCRIPTING DATABASE: This refers to the database SID or global database name (an alias given to a database upon creation).

IES: SCRIPTING BEAN DISPLAY MODE: The Scripting Bean Display Mode parameter must always be set to display the Scripting Java bean in a separate Oracle Form, since the Scripting Engine will always run as a Java bean in a standalone Java-based window. Therefore, for every implementation, ensure this profile parameter is set to the LOV parameter `Display Scripting bean in separate Frame`. ***Do not use*** the LOV option `Embed Scripting bean in form`.

IES: SCRIPTING PANEL DISPLAY MODE: This parameter controls whether one panel is visible at a time in the Scripting Engine, or multiple panels (with the active panel having focus). A value in this parameter is required by default. Select one of the following from the LOV: `Display Multiple Panels at a time in`

`Scripting window`, or `Display Single Panel at a time in Scripting window`.

IES: SERVER HOST NAME FOR SCRIPTING APPLICATION: This parameter either references the database listener host (for Caching Architecture operations) or the IIOP listener host (for 3-Tier JServer Technology Stack operations).

Installation/Implementation Flowchart

The flowchart depicted in Figure 14 below is intended to show all the broad steps required to install and implement Scripting, from Rapid Install, through implementation steps detailed in this document. Note that patches may have prerequisite and required post-patch steps; these would be included in the README.txt file accompanying each patch. It is incumbent upon the person(s) applying these patches to read and follow all steps in each README.txt file unless otherwise indicated by OSS. For the latest information on patches, updates, etc., and to obtain those patches, always refer first to MetaLink.

Implementation Checklists

Following are two complete checklists for implementation steps to be performed on a system that meets the requirements of the Implementation Starting Point above.

The first checklist specifically addresses Caching Architecture implementations only. The second checklist specifically addresses 3-tier JServer Technology Stack implementations, which will be de-supported after April 1, 2001.

Detailed implementation steps follow, tracking to each respective checklist step but providing greater detail.

Oracle CRM Applications support multiple platforms. The checklists below are intended to be applicable to all systems; when information is specific or unique to UNIX or NT systems, this information is denoted. (For more information, see the discussion on platforms in this document; for full details, the list of supported platforms for Oracle Applications can be found in MetaLink.)

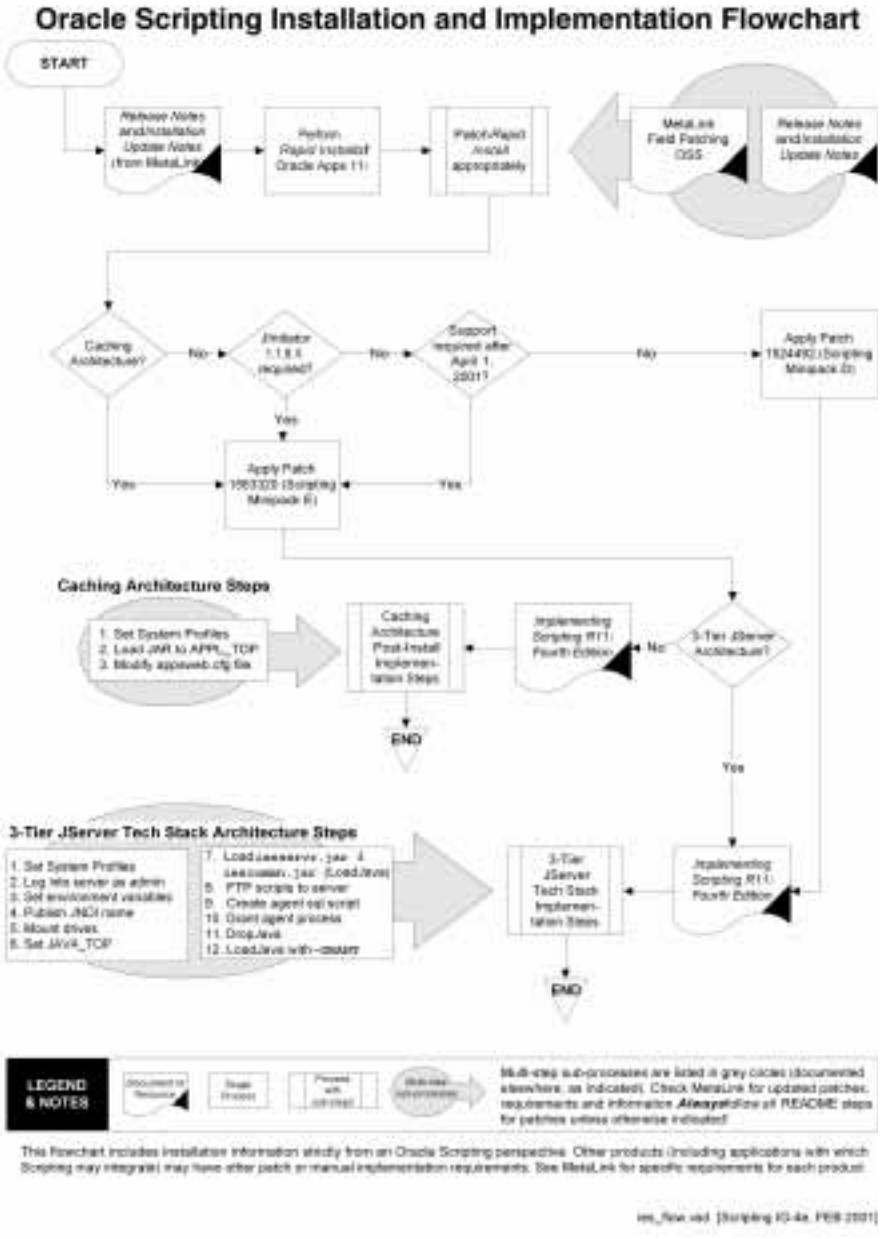


Figure 14. Installation and Implementation Flowchart

Caching Architecture Implementation Checklist

1. For 11.5.2 or later implementations, ensure the Oracle Applications system profile values specific to Caching Architecture operations are appropriately established. For more information, see the settings and their explanations in Setting System Profiles above.
2. Load custom Java code (JAR files) to a directory on the APPL_TOP.
3. Modify the `APPSWEB.CFG` file to appropriately reference custom Java code, the thin JDBC driver, and AOL/J classes.
4. Deploy scripts from Script Author to applications database.
5. Launch scripts (in a demo Oracle Form or from integrated application).

Three-Tier Java Technology Stack Implementation Checklist

1. For 11.5.2 or later implementations, ensure the Oracle Applications system profile values specific to 3-tier JServer Technology Stack operations are appropriately established. For more information, see the settings and their explanations in Setting System Profiles above.
2. Log into server with an administrative account.
3. Set environment variables, including `ORACLE_HOME, PATH, ORACLE_SID, LD_LIBRARY_PATH,` and `CLASS_PATH`.
4. Publish JNDI name.
5. Mount drives (if applicable).
6. Set `JAVA_TOP`.
7. Run LoadJava to move Scripting foundation JAR files (**`iesservr.jar`** and **`iescommn.jar`**) into JServer.
8. FTP the following from Windows NT, 95, 98, or 2000 Script Author workstation to server: (a) SQL script to create agents, and (b) either Shell script (for UNIX server only) or batch (BAT) file (for NT server only) to grant agents privileges to run Java in the database.
9. Run SQL script to create agent user, once *per agent*.
10. Run grant agent shell script (UNIX) or BAT file (NT) one time, to grant agents the ability to run Java in the Oracle 8*i* applications database.
11. Run DropJava (once) and rerun LoadJava (once *per agent*, each time with explicit `-grant` for that agent) to move server-required JAR files

(**`iesservr.jar`** and **`iescommn.jar`**) into JServer while granting specified agents permission to execute the JAR files.

12. Run LoadJava (once *per agent*, each time with explicit `-grant` for that agent) to move custom Java into JServer and grant specified agents permission to execute the JAR files.
13. Deploy scripts from Script Author to applications database.
14. Launch scripts (in a demo Oracle Form or from integrated application).

Detailed Caching Architecture Implementation Steps

Caution: This is a detailed description of the steps listed briefly in the Caching Architecture Implementation Checklist above. *It is not intended for this full list of steps to be repeated.*

Caution: ***Do not perform these steps for 3-tier JServer Technology Stack implementations.*** For 3-tier JServer implementations, refer to the Three-Tier Java Technology Stack Implementation Checklist or the corresponding Detailed Three-Tier Java Technology Stack Implementation Steps.

---

1. For 11.5.2 or later implementations, ensure the Oracle Applications system profile values specific to Caching Architecture operations are appropriately established.

   Specifically, IES: IES ARCHITECTURE TYPE parameter should be set to `Two Tier Mode`; the IES: PORT FOR SCRIPTING APPLICATION parameter should reference the TNS port; and the IES: Server Host Name for Scripting application parameter should reference the database listener host. Other settings are based on the specific requirements of the implementation. For a matrix of all the relevant settings and brief descriptions, see Setting System Profiles above.
2. Load custom Java code (JAR files) to a directory on the APPL_TOP. These should be located in the `ies_custom` directory, as described in Deploying Custom Code to the APPL_TOP.

3. Modify the `appsweb.cfg` file to appropriately reference custom Java code, the thin JDBC driver, and AOL/J classes.

   - **Custom Code:** Any custom Java code (JAR files) used in the Scripting implementation should be referenced in the configuration file to ensure they load when Oracle Applications are launched. The JAR file or files to include in this list would be identical to the ones deployed to the server in the section Deploying Custom Code to the APPL_TOP. For the purpose of continuing our example, this file would be `custom.jar`.

     Add the path and name of the custom JAR file(s) to any location prior to the reference to `fndlist.jar`. Following the example discussed earlier, add:
     `,/OA_JAVA/ies_custom/custom.jar`

   - **JDBC Driver:** Append the path and name of the JDBC driver to any location prior to the reference to `fndlist.jar`.

     While the actual location of this file is not important, it is easier to maintain the configuration file if you keep your modifications close together. Therefore, immediately following the custom JAR files you added in the previous step, add `,/OA_JAVA/jdbc111.zip`

   - **AOL/J Classes** (library functions for making JDBC connections to the database). Anywhere in the file following the reference to `fndlist.jar`, reference the path and name of the AOL/J classes (which are combined into this JAR file) by adding:
     `,/OA_JAVA/oracle/apps/fnd/jar/fndaolj.jar`

     While the actual location of this file is not important, it is easier to maintain the configuration file if you keep related files together. In the example cited in Appendix D, the AOL/J classes are appended to the end of the archive3 block. This archive includes CRM product JAR files that are downloaded by demand to the client (when needed by a particular CRM application).

4. Deploy scripts from Script Author to applications database.

   Follow standard Oracle Scripting rules and precautions when deploying scripts. Keep in mind:

   a. If the script you are deploying is intended to call any custom Java, the script must contain commands that reference any custom Java methods loaded into the JAVA_TOP as indicated above.

   b. A script is identified before deployment and at runtime by two pieces of information entered into the script's global Script Properties: (1) the script name, and (2) the script language. The database disregards any file name

you may give to the script from an operating system perspective. Regarding script language, the value must match an actual value from the NLS_LANGUAGE column of the FND_LANGUAGES table (for example, use "AMERICAN" for American English).

c. Deploying a script with the same name and language as a script that has already been deployed to the same database will cause all earlier versions to be flagged as disabled. Note, however, that this does not delete the old version of the script from the database. For more information, see Using Oracle Scripting > Removing Old Scripts from the Database.

For more information on deploying scripts, see Compiling and Deploying a Script in this document. For detailed information, refer to the Oracle Scripting Concepts and Procedures document.

5. Launch scripts (in a demo Oracle Form or from integrated application).

Scripts are launched from the agent desktop through either the integrated Oracle Forms-based application or through the Demo Form created for this purpose. For step-by-step procedures for launching a script in the Demo Form, see Appendix B. For step-by-step procedures for launching a script from an integrated application, see the relevant documentation for that product or refer to How to Launch Oracle Scripting in this document.

Detailed Three-Tier Java Technology Stack Implementation Steps

Caution: This is a detailed description of the steps listed briefly in the Three-Tier Java Technology Stack Implementation Checklist above. It is not intended for this *full list of steps to be repeated.* Steps required to be performed more than once are indicated below.

Caution: ***Do not perform these steps for Caching Architecture implementations.*** To implement the Caching Architecture from scratch, refer to the Caching Architecture Implementation Checklist or the corresponding Detailed Caching Architecture Implementation Steps. To convert from a 3-tier implementation to the Caching Architecture, see the checklist and detailed steps indicated in Converting Three-Tier Environments to a Caching Architecture below.

1. For 11.5.2 or later implementations, ensure the Oracle Applications system profile values specific to 3-tier JServer Technology Stack operations are appropriately established.

   Specifically, the IES: IES ARCHITECTURE TYPE parameter should be set to `Three Tier Mode`; the IES: JNDI NAME USED FOR SCRIPTING APPLICATION parameter should be set to `/test/oracle/apps/ies/corba/common/Master`; the IES: PORT FOR SCRIPTING APPLICATION parameter should reference the IIOP port; and the IES: SERVER HOST NAME FOR SCRIPTING APPLICATION parameter should reference the IIOP listener host. Other settings are based on the specific requirements of the implementation. For a matrix of all the relevant settings and brief descriptions, see Setting System Profiles above.

2. Log into server with an administrative account.

   NT—Log onto Windows NT server as the administrator.

   UNIX—Log into the host machine where the Applications database was installed. Alternatively, you can log into any UNIX account, assuming you have the privileges and connection information to mount other drives such as the applications server.

   If the database tier for this implementation is installed on a separate server, you must be able to mount both the applications tier server and the database tier server.

3. Set environment variables, including `ORACLE_HOME, PATH, ORACLE_SID, LD_LIBRARY_PATH,` and `CLASS_PATH.`

   `ORACLE_HOME` must be set to the 816 `ORACLE_HOME` environment in which the database exists (i.e., the directory in which the database is installed, regardless of whether it is the same as the applications tier server or on a separate server).

   `PATH` must be set to prepend the `ORACLE_HOME` bin and LIB.

   `ORACLE_SID` is an environmental variable that holds the value of the System Identifier (SID). SID is the name given to an instance / database during its creation. Set the `ORACLE_SID` to point to the applications database instance.

   `LD_LIBRARY_PATH` is the path that includes shared libraries.The syntax for setting the `LD_LIBRARY_PATH` is the same as long as you are running Kornshell (Solaris and other UNIX environments).

   `CLASS_PATH` must be set to the `ORACLE_HOME` bin.

Note: For UNIX, *all environment variables must be set every time you do a Publish, LoadJava, or DropJava.*

4. Publish JNDI name.

   The Java Naming and Directory Interface (JNDI) is an API that provides naming and directory functionality to applications written in the Java programming language. From an Oracle 8*i* database management perspective, publishing the JNDI name (also known as the Master Object) is an implementation step unique to Oracle Scripting implemented in the 3-tier JServer Technology Stack architecture.

   The Scripting client uses the JNDI name to find and instantiate Oracle Applications server objects. Without this step, Scripting would not be able to use its own JAR files on the server or custom Java deployed to the JServer.

   Publishing the JNDI requires the `apps` password, the IIOP port, and the `apps` database instance name.

   Syntax for this command is as follows:

```
 publish -republish -user APPS -password APPS -service
sess_iiop://localhost:<IIOP port>:<SID>
/test/oracle/apps/ies/corba/common/Master
oracle.apps.ies.corba.server.MasterImpl
oracle.apps.ies.corba.common.MasterHelper
```

5. Mount other drives (if applicable)

   Before performing a LoadJava (step 7) to move files from the applications tier to the JServer on the database tier, you must be able to see the source drive (the server containing the compiled Java files you will load into the JServer) and the target drive (the server containing the database tier). For example:

   - If the server you logged into was *not* the applications tier server (where Applications files including **`iesservr.jar`** and **`iescommn.jar`** are installed during Rapid Install); for UNIX, this is the APPL_TOP.
   - If the database tier is on a separate server from the applications tier.

     UNIX—use MOUNT command to be able to see both target and source.

     NT—locate all appropriate drives in the Network Neighborhood, or map both drives in the File Manager.

6. Set JAVA_TOP.

7. Run LoadJava to move Scripting foundation JAR files (**iesservr.jar** and **iescommn.jar**) into JServer. This simply loads the JAR file into the applications schema and has no association with specific agents.

   Syntax:

   ```
   loadjava -user apps/apps -resolve -oracleresolver -synonym
   -definer -oci8 <jar file being loaded>
   ```

   Examples:

   ```
   loadjava -user apps/apps -resolve -oracleresolver -synonym
   -definer -oci8 iescommn.jar


   loadjava -user apps/apps -resolve -oracleresolver -synonym
   -definer -oci8 iesservr.jar
   ```

8. FTP scripts from Script Author workstation to server.

   Two scripts must be moved from the Script Author workstation to the server and executed on the server. These scripts were installed onto the Windows NT, 95, 98, or 2000 workstation during installation of the Script Author by running the Oracle Universal Installer.

   a. The first script is a SQL script to create an agent. Its name is **ies_create_agent_user.sql** and it is located in the Script Author directory on the script development workstation.

      Assuming the Script Author was installed in the Oracle Home directory, the directory path is: [OraHome]\apps\ies\author\server\shellScriptsserver\sqlScripts.

   b. The purpose of the second script is to grant agents the ability to run Java in the Oracle 8*i* applications database.

      UNIX—The grant agent script for UNIX is a shell script named **grant_agent**. Assuming the Script Author was installed in the Oracle Home directory, the directory path is: [OraHome]\apps\ies\author\server\shellScripts.

      NT— The grant agent script for NT is a BAT file named **grant_agent.bat**. Assuming the Script Author was installed in the Oracle Home directory, the directory path is: [OraHome]\apps\ies\author\server\batchFiles.

As a matter of course, after moving the files via FTP to the server, verify that they transferred correctly. On the server, navigate to the directory to which you moved the files, and enter the list files command, looking for the appropriate file names as listed above.

9. Run script to create agent user.

Run **`ies_create_agent_user.sql`** script on server using SQL*Plus (or other SQL tool) to create an agent.

Note: ***This process must be performed for each agent in the call center.*** There is currently no batch process to perform this multiple times. This must be made clear to the DBA of the implementing enterprise if you are not setting up all agents at implementation.

---

10. Run grant agent shell script (**UNIX**) or BAT file (**NT**) once *per agent login* to grant agents the ability to run Java in the Oracle 8*i* applications database.

Note: This process need only be performed once for each new Scripting username and password that you created in step 9.

Note: Running this shell script / BAT file *may* load two JAR files (`common12.jar` and `server12.jar`) or indicate an error stating that the files are not present. These files are not needed; if present, they may be deleted. If an error message is received to this effect, the message can be ignored.

---

11. Run DropJava one time per JAR file to drop the specified Java classes (`iesservr.jar` and `iescommn.jar`) from the JServer. Then reexecute LoadJava (once *per agent*, each time with explicit `-grant` for that agent) for the Scripting-specific JAR files (`iesservr.jar` and `iescommn.jar`).

When running LoadJava with `-grant`, you are loading the specified JAR file in the applications schema *and* granting access to the specified agent user to execute that Java.

Syntax:

```
dropjava -user apps/apps@<SID> -resolve -oracleresolver
-synonym -definer -oci8 <jar file being dropped>

loadjava -grant <agent_name> -user apps/apps@<SID> -resolve
-oracleresolver -synonym -definer -oci8 <jar file being
loaded>
```

Examples:

```
dropjava -user apps/apps@<SID> -resolve -oracleresolver
-synonym -definer -oci8 iescommn.jar

dropjava -user apps/apps@<SID> -resolve -oracleresolver
-synonym -definer -oci8 iesservr.jar

loadjava -grant ies_agent -user apps/apps@iem11icw -resolve
-oracleresolver -synonym -definer -oci8 iescommn.jar

loadjava -grant ies_agent -user apps/apps@iem11icw -resolve
-oracleresolver -synonym -definer -oci8 iesservr.jar
```

Note: The requirement to run DropJava prior to running Load-Java a second time has been introduced since the last edition of this Implementation Guide. Including this clarification step will avoid occasional problems previously encountered in the database.

---

12. Run LoadJava (with `-grant`) to move custom Java into JServer.

Following the example above, you must run LoadJava to move any custom code into the database.

When deploying custom compiled Java to the JServer, you *must include explicit* `-grant` *privileges* for all agents required to run the script, as described in step 11 above. If you exclude this step, the custom Java can only be called by an agent using a Scripting login associated with `apps` privileges.

If you have already deployed that custom code, run DropJava first, as in the example above. Skipping this step can result in unexpected errors in the Scripting environment.

Caution: When running DropJava, ***ensure that you specify by name which code you want to drop*** (as in the example in step 11 above), or *all* Java may be dropped from the JServer.

Note: Never run DropJava with `-grant`.

13. Deploy scripts from Script Author to applications database.

Follow standard Oracle Scripting rules and precautions when deploying scripts. Keep in mind:

a. If the script you are deploying is intended to call any custom Java, the script must contain commands that reference any custom Java methods loaded into the JServer in step 11.

b. A script is identified before deployment and at runtime by two pieces of information entered into the script's global Script Properties: (1) the script name, and (2) the script language. The database disregards any file name you may give to the script from an operating system perspective. Regarding script language, the value must match an actual value from the NLS_LANGUAGE column of the FND_LANGUAGES table (for example, use "AMERICAN" for American English).

c. Deploying a script with the same name and language as a script that has already been deployed to the same database will cause all earlier versions to be flagged as disabled. Note, however, that this does not delete the old version of the script from the database. For more information, see Using Oracle Scripting > Removing Old Scripts from the Database.

For more information on deploying scripts, see Compiling and Deploying a Script in this document. For detailed information, refer to the Oracle Scripting Concepts and Procedures document.

14. Launch scripts (in a demo Oracle Form or from integrated application).

Scripts are launched from the agent desktop through either the integrated Oracle Forms-based application or through the Demo Form created for this purpose. For step-by-step procedures for launching a script in the Demo Form, see Appendix B. For step-by-step procedures for launching a script from an integrated application, see the relevant documentation for that product or refer to How to Launch Oracle Scripting in this document.

Converting Three-Tier Environments to a Caching Architecture

The following checklist indicates steps required to convert an existing 3-tier JServer Technology Stack environment to the Caching Architecture. Following are detailed steps describing the specifics of each process. Note that Scripting Minipack E (patch 1563320) is a prerequisite of Caching Architecture operations.

Three-Tier JServer Technology Stack to Caching Architecture Conversion Checklist

1. Load all custom Java code (previously deployed to the JServer) to the appropriate directory on the APPL_TOP.
2. Modify the `appsweb.cfg` file to appropriately reference custom Java code in the core classes.
3. Modify `appsweb.cfg` file to include the JDBC driver in the core classes.
4. Modify `appsweb.cfg` file to include AOL/J classes in the load-on-demand archive3 block.
5. For 11.5.2 or later implementations, change the relevant Scripting Application Profiles to parameters specific for Caching Architecture operations.
6. Execute scripts in the Caching Architecture.

Detailed Three-Tier JServer Technology Stack to Caching Architecture Conversion Steps

1. Load all custom Java code to the appropriate directory on the APPL_TOP.

   While in the 3-Tier JServer Technology Stack, compiled `.class` files could be deployed to the JServer, the Caching Architecture requires custom Java to be deployed in Java Archive (`.jar`) file format. Note that multiple `.class` or `.jar` files can be combined into a single `.jar` file.

   In 3-Tier JServer Technology Stack operations, each custom Java file needed to be deployed to the JServer using LoadJava and with explicit `-grant` command executed for *each agent required to run Scripting*. This is not required in the Caching Architecture.

   The appropriate directory is `$JAVA_TOP/ies_custom`, located in the JAVA_TOP directory under the APPL_TOP. For specific instructions, see Using Oracle Scripting > Deploying Custom Code to the APPL_TOP above.

2. Modify the `appsweb.cfg` file to appropriately reference custom Java code in the core classes.

The core classes are located prior to the reference to `fndlist.jar`. Using the sample `appsweb.cfg` file in Appendix D as an example, custom code would be referenced at the end of the first archive block (immediately preceding reference to the `fndlist.jar`). For more information, see Appendix D.

This is also described in detail in step 3 of Detailed Caching Architecture Implementation Steps above.

3. Modify `appsweb.cfg` file to include the JDBC driver in the core classes.

   For more information, see step 3 of Detailed Caching Architecture Implementation Steps above or Appendix D below.

4. Modify `appsweb.cfg` file to include AOL/J classes in the load-on-demand section (following reference to the `fndlist.jar` file).

   For more information, see step 3 of Detailed Caching Architecture Implementation Steps above or Appendix D below.

5. For 11.5.2 or later implementations, revise relevant Scripting Application Profile values.

   Replace 3-tier JServer Technology Stack parameter values (represented by values in the second column in the table below) with the appropriate Caching Architecture values (shown in third column).

   Some procedural steps are included in Setting System Profiles in this document. For more information, refer to the appropriate Oracle documentation for administering Oracle Applications.

Profile Value:	REPLACE:	WITH:
IES: IES ARCHITECTURE TYPE:	Three Tier Mode	Two Tier Mode
IES: Port for Scripting application	<IIOP Port>	<TNS Port>
IES: Server Host Name for Scripting application	<IIOP listener host>	<Database listener host>

6. Execute scripts in the Caching Architecture.

   Since these steps are for converting existing 3-tier JServer environments to the Caching Architecture, scripts should already be deployed to the JServer. If modifications were made, open the Script Author, make modifications, and redeploy the relevant script to the database.

Testing Oracle Scripting

Implementation Testing Considerations

There are three distinct but related considerations for testing an Oracle Scripting implementation:

1. Manual post-installation implementation steps (performed after Rapid Install or other installation of Oracle Applications) must be tested.
2. Any custom scripts developed for use by the implementing enterprise must be tested.
3. Any integration between Oracle Scripting and other applications (CRM 11*i* business applications, ERP applications, or custom legacy applications) must be tested.

These three considerations are discussed below as the basis for this guide's coverage of testing Oracle Scripting. For business requirements and resource planning purposes, note that different individuals or teams may be required to address each of these three considerations. For more information on business requirements specification and methodology recommendations, see AIM's Role in Scripting Implementation Planning or AIM and Implementation Testing in this document.

Generic Steps for Testing Any Script

In order to test that Scripting has been installed successfully at a customer site, you must follow these steps:

1. Create a new script or open an existing script using the Script Author.
2. Ensure that the script in question is syntactically correct and can compile.
3. Deploy the script to the Oracle 8*i* applications database.

4. If there is any custom Java code supporting the script deployed, it must be deployed to the server.

   Caching Architecture: Requires `.jar` files only, deployed to the APPL_TOP, and requires those `.jar` files to be referenced in the `appsweb.cfg` file on web server. These are described in Using Oracle Scripting > Deploying Custom Code to the APPL_TOP above.

   3-tier JServer Technology Stack: Requires `.jar` (recommended) or `.class` files, deployed with -grant to the JServer, as described in step 11 of Implementing Oracle Scripting > Detailed Three-Tier JServer Technology Stack Implementation Steps.

5. Log into Oracle CRM 11*i* applications with the appropriate responsibility.
6. Launch Scripting from the Demo Form or appropriate integrated application.
7. Run through the script to ensure the Scripting Engine and the script being executed are functioning properly.

Testing Manual Post-Installation Implementation Steps

To test that Oracle Scripting has been successfully installed and implemented (including all manual post-installation step), you must:

1. **Deploy and test a simple script with no associated custom Java.** Step-by-step instructions to create, modify, deploy and run a simple script are included in Appendix B of this document.
2. **Deploy and test a script with custom Java code.** For this purpose we specifically recommend using the Regression Test Script discussed in the the Related Documentation and Resources section below. Alternatively, you can use the custom script specifically developed for this enterprise, assuming it has already been debugged and executed successfully in a working environment. However, this may result in it being more difficult for you to isolate any problems you might encounter, and lead to misdiagnosis of those difficulties. Deploying and testing the Regression Test Script obviates these concerns.

Why Test a Script With No Java?

The first test script to be deployed and executed should have no custom Java methods associated with it. In this way you quickly verify that:

- A script can be launched in the designated environment, independent of the technical considerations related to custom Java.
- For 3-tier JServer Technology Stack implementations, at least one agent is appropriately set up, including an appropriate Scripting database login, and

privileges are appropriately granted for agents to execute Scripting product-specific JAR files.

- You eliminate certain environmental variables as the cause of problems if this test succeeds and the subsequent test (launching and executing a script that calls custom Java methods) fails.

Specifically, you have tested the following successfully:

- The Oracle Universal Installer process to load the Script Author on a client workstation has been successful.
- Thin JDBC works appropriately from the Script Author client workstation and the database is functioning at the time of testing.

Additionally, if implementing in the 3-tier JServer Technology Stack, testing a script with no Java also verifies that:

- The JNDI name has been appropriately published.
- The IIOP port is successfully administered and is using the IIOP protocol to instantiate CORBA sessions.
- The `ies_create_agent_user.sql` script has been run to successfully create the agent login for the agent or agents used during testing.
- The `grant_agent` process has been successfully executed.
- The Scripting-specific JAR files—`iescommn.jar` and `iesservr.jar`—have been appropriately loaded into the JServer with `-grant` explicitly for the agent used during testing.

Once the functionality of scripting has been tested, custom scripts must be deployed and tested, followed by specific integration testing. Based on contracted services, project phasing and scoping, customized script testing and any integration testing may be performed by different individuals or teams. Reference the appropriate documentation for your project (e.g., AIM CR.010, TE.040, etc.).

Why Use the Regression Test Script?

You need functioning scripts to test a newly implemented environment, and you needs a functioning environment to test scripts. To avoid a “chicken and egg“ situation, we recommend deploying and testing a ready-made Regression Test Script (along with its associated custom Java files). See Regression Test Script in this document for information on obtaining this script.

If it is your responsibility to test not only whether Scripting has been appropriately installed, implemented and configured, but also to test whether a script customized for that enterprise will function, you will be tempted to take a shortcut and skip the Regression Test Script. However, this may result in it being more difficult for you to isolate any problems you might encounter that are related not to the Scripting implementation in general but to the enterprise-specific requirements, environmental variables, or the custom script. Deploying and testing the Regression Test Script obviates these concerns.

If you can successfully deploy and test the Regression Test Script, you quickly verify that:

- A script with custom Java methods associated with it can be launched in the designated environment.
- Pending the appropriate configuration of environmental variables, a customized script will function properly in this environment.

For Caching Architecture environments, successful execution of the Regression Test Script also signifies:

- You have appropriately loaded the custom JAR file to the APPL_TOP;
- You have appropriately modified the `appsweb.cfg` file to reference custom code; and
- The custom code is successfully cached to the client by JInitiator.

For 3-Tier JServer Technology Stack environments, successful execution of the Regression Test Script also signifies that:

- Java is appropriately running in the JServer of the database tier.
- IIOP is successfully configured on the database.
- The `.class` or `.jar` file for the Regression Test Script Scripting has been appropriately loaded into the JServer with `-grant` explicitly for the agent used during testing.

Who Performs Manual Post-Installation Implementation Testing?

Testing of manual post-installation implementation steps listed in this document (through executing and testing a simple script and one with custom Java such as the Regression Test Script) is typically be performed by the individuals or team that implemented those post-install steps.

Customized Script Testing

Scripts developed by OCS or Oracle partners and delivered and implemented at customer sites are customized to meet the customer's specific requirements. Following successful execution of test scripts with and without custom Java code, the customized script and custom components supporting the script must be tested. This may be performed by different individuals or teams than those who tested that the product has been successfully installed and implemented.

Prior Business Planning Required

The process of defining a script for a customer implementation should follow the conventions of understanding the customer's specific business needs, defining the requirements for the script, designing the script, setting customer expectations, and then creating and testing the script. For more information, see Planning an Oracle Scripting Implementation in this document.

Who Performs Customized Script Testing?

Testing of custom scripts developed for the implementing enterprise is typically performed by the individuals authoring the custom script and those developing supporting custom Java methods, PL/SQL statements, Forms commands, plug-in Java Beans, or other components.

In order to test that custom scripts and all their components have been implemented successfully at a customer site, you must follow these steps:

1. Test the custom script and associated Java methods in a proven environment (at the consulting organization's home base or test lab).

   Note that some elements of a script—particularly portions that integrate with legacy systems, databases, and proprietary software—will not be able to be tested "in a vacuum." Placeholders will have to be developed in the script as workarounds, along with custom Java methods to simulate data or events the script will encounter when finally deployed to its "native" environment.

2. Copy the customized script developed specifically for this customer implementation to the properly configured client machine hosting the Script Author.
3. Copy the custom Java files, PL/SQL commands, and other components developed specifically for this customer implementation.
4. Ensure that the script in question is syntactically correct and can compile.
5. Deploy the custom script to the implementing enterprise's applications server.

6. For 3-tier JServer Technology Stack implementations, perform LoadJava (with `-grant`) to load custom `.class` or `.JAR` Java files into the JServer of the implementing enterprise's Oracle 8*i* applications database tier.
7. Move other custom code to the appropriate implementing enterprise's servers (install PL / SQL cartridges, etc.).
8. Log into Oracle CRM 11*i* applications with the appropriate responsibility.
9. Launch Scripting from the Demo Form or appropriate integrated application. For 3-tier JServer Technology Stack implementations, this will include the additional step of performing a Scripting login to the database.
10. Run through the script to ensure the Scripting Engine and the script being executed are functioning properly and to eliminate environmental variables as failure points.
11. Modify functioning script to integrate customized portions that could not be tested previous to deployment in customer's environment.

    Following successful execution of customized script in the enterprise's environment, modify the script incrementally to test each portion as you add custom components, to minimize debugging and integration issues. Keep careful notes and back up each version of the functioning script and any custom code.
12. Execute, debug and modify script as appropriate until expected behavior results.
13. Test system according to the test plan specifications agreed to between OCS or the Oracle Partner performing the implementation and the enterprise.

Script Integration Testing

If you have successfully performed testing of the Scripting implementation and have tested the customized script for the implementing enterprise, you should be able to test scripting integration by following these steps:

1. Launch the business application using an agent with the appropriate responsibility for that application and who has a valid Scripting database authentication login.
2. Meet the requirements for that application; most likely, this means having a customer record selected prior to launching the script.
3. Initiate the script in the prescribed manner for that product (for instance, in TeleService, click the Script icon from the toolbar; from TeleSales, click the View Script button).

4. For 3-tier JServer Technology Stack implementations, a second login is required. Enter the appropriate Scripting login information for that agent.
5. Progress through the script following the designated test plan for this particular implementation and resolve any problems with the custom script as appropriate.

For more information on recommendations regarding using Scripting in context with other applications, see Scripting Is Not Intended For StandAlone Use in this document.

Who Performs Integration Testing?

Integration testing for Oracle TeleSales or Oracle TeleService is typically performed by those testing the customized script. If integrating with other products, legacy systems, custom tables, etcetera, those who perform specific integration steps will typically test them. In this case, it is recommended that any test plans created by the customer or the script customization team be retested upon integration. Additional test cases may be required. The testing phase should have been carefully considered prior to implementation, and acceptance criteria should be reviewed by management of the implementing enterprise and by management of each team performing implementation.

eBusiness Products With Which Oracle Scripting Integrates

Oracle Scripting can be integrated with the Oracle Customer Support module of Oracle TeleService, as well as Oracle TeleSales, Oracle Banking Center, Oracle iSupport, and Oracle Collections Center. Please refer to the most recent documentation for each individual product listed above regarding integration methods and the degree of integration with Scripting. For more information on Scripting integration, see Scripting and Integration of Other CRM Applications in this document.

Forms-Based applications

Scripting is designed to work seamlessly with Forms-based applications. A few CRM 11*i* applications include integration with Scripting out of the box.

Custom Forms-based applications can also be integrated with Scripting. The Scripting UI includes several individual components that can be included in the application, via the Bean Area of Forms 6.0, as well as a Frame that includes all of the individual components in a default layout.

In addition, Scripting provides two sets of APIs for external applications to manipulate a running script. First, the Proxy Bean exposes APIs for manipulating

the Scripting server object, and second, the Client Engine object allows an application to instantiate and manipulate the UI components of the Scripting product. For more details, refer to the *Oracle Scripting External APIs and Events* document.

For 3-tier JServer Technology Stack implementations, note that the Applications login—the login / password combination used for logging into Forms-based applications—cannot be used to authenticate a Scripting session. The login used to authenticate a Scripting session must be an agent database login (created as described in the Post-Installation steps for Agent Creation).

For more information on Scripting integration, see Scripting and Integration of Other CRM Applications in this document.

CTI Integration

In the long term, Scripting 11*i* will integrate with CTI. This takes three possible approaches:

1. Integrated with a CRM 11*i* business application (Customer Support, TeleSales, etc.) In this scenario, the business application handles the telephony integration. It communicates to Oracle Universal Work Queue (UWQ) and gets the script to do the appropriate thing at the appropriate time. Scripting doesn't know or care about the telephony functions at all.
2. Scripting standalone integrated to CTI in CRM 11*i*. At the time of Oracle Applications release 11.5.2, Oracle Scripting does not integrate out-of-the-box with UWQ. However, this functionality is expected in a future release. The integration will include a “pop“ of the script from UWQ based on the agent’s selection of a task that invokes the Scripting Engine.
3. Scripting standalone integrated to other CTI suites. At the base level, we have attempted to make Scripting completely flexible so that it can be used in any implementation and doesn't lose any useful functionality. To achieve this, we have implemented two distinct APIs: one is non-GUI specific and allows any client to manipulate our server functionality (the Proxy Bean). The second is GUI specific (the Client Engine) and allows any custom application to manipulate our default GUI (and get access to the Proxy Bean for server manipulation). With these two sets of APIs, a consultant with Java programming experience can build a custom shell for his client that will integrate with the third party CTI vendor and will also embed and manipulate the Scripting functionality. For example, we have a simple API for starting a script: `startScript` takes two arguments, name of the script and language. The custom application needs to get the incoming call event from the third party CTI software and then call `startScript` to bring up the right script.

For More Information on Scripting Integration

Integration of Scripting and other Oracle products is a large topic that could encompass a discussion of various technologies (Java, Java Beans, Forms, CORBA integration, screen-scraping, etc.). As such, there are a variety of topics that are outside the scope of this document. Nonetheless, in the paragraphs above, an attempt has been made to include helpful information for those who need to integrate Scripting.

It is also recommended that you carefully review the available documentation for applications you must integrate with Scripting, particularly references to existing application program interfaces (APIs).

For late-breaking information on the latest Oracle Scripting implementation and integration success stories, those with access to the Oracle Intranet should monitor

Prerequisites

Prior to testing a scripting implementation, the customer's environment must prove to be stable. The network and the database must be up and running, the "Rapid Install" must be accomplished, all appropriate patches and their requisite manual steps must be applied, and all PL/SQL packages tested independently prior to beginning system testing.

Troubleshooting Oracle Scripting

Troubleshooting—Enabling Database Trace Files

All database transactions for a particular user can be captured in database trace files. Trace files are automatically generated and stored in a location determined by the database configuration and the operating system. Obviously, considering how many transactions may occur, these trace files can be large in size. For this reason, the database may be set up to discard trace files by default, or only save trace files of a certain size. These options are typically configured at installation.

Working with Oracle Scripting, it is important to be able to view trace files to help debug PL/SQL code (used in script Commands), debug failed SQL calls (such as from a Block in a script), or determine other database issues.

Enabling database trace files is even more important for those implementing Scripting on a 3-tier JServer Technology Stack, because *all Java output* is also written to database trace files when using this architecture. (This output is written to the JInitiator Java Console in Caching Architecture implementations. For more information, see Related Products and Components > Oracle JInitiator > Steps to Enabling the Java Console.) Thus, for 3-tier JServer Technology Stack implementations, consult trace files not only to debug PL/SQL code or failed SQL calls, but also to debug custom Java code, verify Java in the database, view CORBA or IIOP error messages, etc. Viewing trace files is particularly important during implementation of a 3-tier JServer Technology Stack environment.

Tracing can be enabled specifically (per agent, per session) by a qualified database administrator. Tracing may already have been enabled in your environment (see next section). To enable tracing, check with the enterprise or implementation DBA.

Locating Default Trace File Locations (UNIX and NT)

The following information describes how to locate background trace files (trace files containing Java output). Consult a DBA to determine how to locate user trace files (containing SQL and other output).

1. To locate the default trace files location, log into the database using a SQL tool such as SQL*Plus or SQL*Worksheet.
2. Execute the following command:

   ```
   select value from v$parameter where name = 'background_
   dump_dest'
   ```

   UNIX - This will return a UNIX directory where trace files are stored; for example: `/u01/oracle/admin/<instance name>/bdump`

   NT - This will return an NT directory where trace files are stored; for example: `c:\orant\admin\db816nt\bdump`

Other References Regarding Setting Trace Files

For information on setting trace files, refer to the *Oracle 8i Database Administration Guide* for details.

Troubleshooting—Applet Security and Signing

A yellow bar appearing on the bottom of all Forms windows that reads `Warning: Applet Window` indicates that either the Applications JAR files have not been signed properly, or more likely, that the proper security certificate has not been installed on the client JInitiator. For more details on proper security configuration, consult your Oracle Applications installation documentation.

Improper security configuration can lead to a variety of problems in Oracle

Scripting. Among them, the ability to copy and paste to and from fields in Scripting will be disabled. Be sure to properly configure applet security on each client machine.

Troubleshooting—Configuring IIOP Listener

This section is relevant to 3-tier JServer Technology Stack implementations only.

When configuring the IIOP Listener (performed by a DBA), ensure the host name of the machine is used, not just the IP address. The first time the listener is used, the IP address and hostname will be called, but subsequently *only the host name is used.*

This is specified due to the method used in the setServerAPI listed in the API document.

Troubleshooting—Starting Point of 11.5.1 Rapid Install In Lieu of 11.5.2

The main patches required for the Rapid Install process (if compared to the internally available CRM Field Patching web site) are identical for Rapid Install 11.5.1 and 11.5.2. However, some differences should be noted, mostly due to patches required for 11.5.1 installations having been rolled into the 11.5.2 release. This does not purport to be a comprehensive list; be sure the read all the README files in each patch to ensure you have the right files.

If implementing from an 11.5.1 Rapid Install:

- The CRM Family Pack 1 Patch 1306413 and its requisite post-patch steps will be required immediately after performing the manual post-install implementation steps detailed in this document.
- Following Patch 1306413, apply Order Management Family Pack Patch 1288742.
- Include any requisite applications patches *other than* those required for call center applications. All call center / interaction center product patches, including Scripting-only RUP 3, have been rolled up into RUP Patch 1475146, included in the flowchart.

Troubleshooting—Rapid Install Not Used as Starting Point

This section is relevant to 3-tier JServer Technology Stack implementations only.

The Rapid Install process—the recommended starting point for implementing Oracle Scripting—lays down the Oracle 8*i* database as well as all required applications.

As part of this process, Java is configured to run as the Java Virtual Machine (JVM) within the database. Oracle Scripting relies on this Java server embedded within the 8*i* database (known as the JServer).

If for some reason a Scripting implementation is proceeding from other than the recommended starting point, it is crucial that Java be enabled within the database. Equally important is the appropriate setting up of IIOP in the database. For these tasks, a DBA with specific knowledge of Oracle 8*i* database management is essential. See Implementation Roles and Functions in this document.

Setting Up Java in the Database

In the event Java is not enabled in the database (for 3-tier JServer Technology Stack implementations), Scripting will not function. Indications that Java is not enabled in the database include (but are not limited to) the following:

- `initjvm.sql` script indicates errors
- Failure to publish JNDI name properly.
- No `org/omg/CORBA/ORB` class in the database according to the trace files

Failure of initjvm.sql

The `initjvm.sql` script (mentioned in Note 69043.1 on MetaLink, under the section about configuring MTS) describes how to configure the IIOP Port. For implementations of Oracle Scripting in which the Rapid Install was not the starting point, the initjvm.sql script may have been used.

In the event the script fails, it does not create the appropriate ORB objects and initialize the JServer properly. In this case, execute the script again so the ORB and CORBA objects start. This task should be performed by a qualified DBA.

Failure to Publish JNDI Name Properly

Publishing the Java Naming and Directory Interface (JNDI), a 3-tier JServer Technology Stack Scripting-unique requirement for Oracle 8*i*, requires Java to be enabled within the database. This step needs be applied a single time on the CRM 11*i* applications server following the Rapid Install. See Oracle Scripting CRM Rapid Install Manual Post Install Processes.

Missing org/omg/CORBA/ORB Class

If the `org/omg/CORBA/ORB` class is missing in the database according to database trace files, it is possible the JNDI has not been properly published or that Java has not been enabled in the database. Seek the assistance of a qualified DBA experienced with Oracle 8*i*.

If the Java error returned is: `JAVA.LANG.NOCLASSDEFFOUNDERROR: ORG/OMG/CORBA/ORB WHEN CONNECT USING IIOP`, this may be caused by the absence of some `vbj*.jar` files in $ORACLE_HOME/lib. To address this, refer to Note 120227.1 in MetaLink.

This document does not purport to be a treatise on Oracle 8*i* administration. Please refer to appropriate existing documentation for more information. A more complete set of references is available in the section below named Related Documentation and Resources.

Additional References

- *Oracle 8i Administrator's Guide*
- *Oracle 8i Enterprise Java Beans and CORBA Developer's Guide*
- *Oracle 8i Java Developer's Documentation Guide*
- Latest postings on MetaLink

Troubleshooting—Testing an Implementation Project

Scripting Variables

When testing a scripting application, it is important to ensure each component is functioning on its own and in consonance with all others. From the standpoint of the script itself, one must test:

- The call guide or script (the document developed with Script Author)
- All custom Java methods developed to support the script
- All PL/SQL procedures attached as commands to panels, blocks, branches, etc.
- All blackboard commands
- All Forms commands
- All constant commands
- All delete actions

Environmental Variables

One potential pitfall is the complexity of the scripting and Oracle applications environment. For example, even if all aspects of the script are functioning, the following are potential points of failure:

- Network
- Database server
- Web server
- PL/SQL package
- Granted user roles/privileges attached to logged in agent or scripting user (DBA level)

Troubleshooting—Deploying a Script

Problem: Deploying a script to the database with the Author fails.

Solution: The tablespace for the `IES_DEPLOYED_SCRIPTS` table may not be large enough. Particularly indicative is the scenario in which previous scripts were deployed to the database without difficulty. To resolve, increase the size of the tablespace (usually named `USER_DATA`) to an appropriate size (based on database usage and other installed applications). Consult 8*i* documentation for details.

Troubleshooting—Launching a Script for the First Time

A JInitiator JCache directory is required on the client prior to logging into Oracle Applications. If missing, the Java console will indicate errors such as:

```
Default cache directory [Drive]:\Program Files\Oracle\
JInitiator [version]\jcache not found. JAR caching disabled.
```

Ensure that the appropriate version of JInitiator is downloaded and installed on each client prior to running Oracle Applications and that the JCache directory exists in the required path. For more information about Oracle JInitiator, see Related Products and Components > Oracle JInitiator.

Troubleshooting—Launching a Script

Problem: When running embedded within a Forms 6.0 form, the drop-down list of a drop-down control does not appear.

Solution: Contact Oracle Support for the patch to bug #1149726.

Problem: In 3-tier JServer Technology Stack environments, an error occurs while running the script: the Java console doesn't have much error information to help debug the error condition.

Solution: Look at the trace files on the database server. It may be useful to clear the contents of the directory, and reproduce the problem. This will leave only one trace file in the directory and no confusion as to which trace file is the one that documents the problem. For more information, see TroubleShooting—Enabling Database Trace Files above.

Troubleshooting—Configure DB Settings for Expected Load

This section is relevant to 3-tier JServer Technology Stack implementations only.

The following is the list of parameters in the database's `init.ora` file that should be configured for scripting:

shared_pool_size = 50 MB

- java_pool_size = (1 - 5) MB * number of clients [depending on size of script, 1 for small script, 5 for very large script]
- sessions = 50 + number of clients
- processes = 50 + number of clients
- mts_servers = number of CPU s
- mts_max_servers = <mts_servers> * 2
- remove large_pool if exists

> **Note:** These are recommended settings, assuming only Scripting is being run. These numbers should be used as guidelines for extrapolating actual settings for a customer installation, with guidance from an experienced DBA who is familiar with the customer installation.

Troubleshooting—Performance Issues

Speed of Loading Oracle Scripting

There is a method of modifying the `appsweb.cfg` file to preload Scripting-specific JAR files, decreasing the amount of wait time for Scripting to instantiate by eliminating a JInitiator time-stamp comparison. See Appendix D > Pre-Loading Scripting `.jar` Files to Reduce Scripting Startup Time for more information.

Performance Impact After Initial Execution

If experiencing reduced performance subsequent to executing a script the first time, the DBA should use the SQL command ANALYZE, which can optimize SQL queries using the Cost-Based Optimizer. Information on this utility is found in:

- *The Oracle 8i SQL Reference*
- The *Oracle 8i Designing and Tuning Performance* reference
- Chapter 5 of the *Oracle 8i Java Developer's Documentation Guide*

References

For each of the documents referenced in *Performance Issues* above, search the Oracle Documentation Library.

Related Documentation and Resources

The following is a collection of related documentation and resources for additional information on Oracle Scripting and underlying or supporting technologies. If the information cited below was not included with your product, check MetaLink or contact your OSS representative to determine if the information can be provided to you.

Documentation for the Oracle Applications eBusiness Suite is available in Adobe Acrobat® format. You may view the documentation as created using the Adobe Acrobat viewer, which is available free of charge directly from Adobe Corporation. You may also purchase printed, perfect-bound versions of the documentation online via the Oracle Store. The documentation listed here was current as of December, 2000. It is advisable to check MetaLink for any new or updated documentation.

> **Note:** For those with access to the Oracle intranet, hyperlinks to these documents are provided. Note that the exact links are subject to change as documents are updated. Thus, it may be necessary to back out to a higher level directory in the web link provided to find a document if its name has changed.

Oracle Call Center Applications Setup Guide This guide covers the installation of Oracle Call Center and Telephony Applications components including the Oracle Scripting Script Author.

Oracle Scripting Concepts and Procedures 11*i* The *Concepts and Procedures* Guide provides information and instructions to help you work effectively with Oracle Scripting.

Installing Oracle Applications 11*i* Installing Oracle Applications provides instructions for managing your installation of Oracle Applications products. In this release of Oracle Applications, much of the installation process is handled using the new Oracle Rapid Install product, which minimizes the time it takes to install Oracle Applications and the Oracle 8*i* Enterprise Edition technology stack by automating many of the required steps.

Supplemental CRM Implementation Steps The *Supplemental CRM Implementation Steps* guide (formerly *Implementing CRM Applications)* provides instructions for completing an installation of Oracle Customer Relationship Management (CRM) products following a Rapid Install. It addresses manual post-install implementation steps such as publishing the JNDI, loading application JAR files in the applications database, and applying relevant patches. However, these steps are more completely documented in this implementation guide (see the Three-Tier Java Technology Stack Implementation Checklist), and apply *only* to 3-tier JServer Technology Stack implementations. As always, check MetaLink for any updates, new requirements, patches, etc.

Oracle 8*i* Administrator's Guide This is a useful DBA reference that is also available under the umbrella of the Oracle Documentation Library referenced below. Helpful information relevant to this Implementation Guide includes information on creating and configuring databases, setting trace files, Configuring Java Option Connections, the IIOP listener for CORBA, etc.

Oracle 8*i* Java Developer's Documentation Guide The *Oracle 8i Java Developer's Guide* contains topics such as debugging server applications and schema object tools such as LoadJava and DropJava.

Oracle 8*i* Enterprise Java Beans and CORBA Developer's Guide
Holds important DBA reference information such as how to set up the IIOP Listener and Dispatcher.

Oracle 8*i* SQL Reference This document is just what it advertises.

Oracle 8*i* Designing and Tuning for Performance This document enables you to enhance Oracle performance by adjusting database applications, the database, and the operating system.

Oracle Applications Concepts This document describes Oracle Applications concepts such as architecture, supported languages and internationalization, the

filing system (including the APPL_TOP discussed in this document), environment settings, distributed architecture, and more.

Oracle Applications Release 11.0.3 Interoperability Patch for Oracle 8*i* Server Release 8.1.6 This document describes how to set up Oracle Applications Release 11.0.3 (or higher) to run with Oracle 8*i* Server Release 8.1.6. This includes instructions for applying the interoperability patches required for installation of Oracle Applications.

Oracle Documentation Library This library contains a vast subdirectory of useful links. At the top level, these include: Oracle 8*i* Server and Data Warehousing, Oracle8i Server Application Development, Oracle 8*i* Server Networking and Security, Oracle8i Parallel Server, SQL*Plus, Oracle 8*i* interMedia, Spatial, Time Series, and Visual Information Retrieval Options, Oracle 8*i* Java Developer's Documentation, Oracle WebDB, Oracle 8*i* Readme, and Oracle Parallel Fail Safe.

External APIs and Events Design Specification for Oracle Scripting This *Oracle internal-use-only document* details the external interfaces that Oracle Scripting provides in order to allow external applications to include and extend Scripting functionality within the application. (If link no longer functions, refer directly to Oracle Scripting web site.)

Oracle Applications Documentation Library This library contains a wide variety of useful resources including the *Oracle 11i Concepts* document and *Release 11.0.3* and *Oracle 8i Release 8.1.6 Interoperability Patch Notes.*

A

Acronym Glossary

The following acronyms are used throughout this document:

AIM	Application Implementation Method
API	Application Programming Interfaces
ARU	Automated Release Update (e.g., a patch)
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
CTI	Computer-Telephony Integration
DBA	Database Administrator
ERP	Enterprise Resource Planning
GUI	Graphic User Interface
IDE	Integrated Development Environment
IG	Implementation Guide
IIOP	Internet Inter-ORB Protocol
JAR	Java Archive
JCK	Java Compatibility Kit
JDK	Java Development Kit
JDBC	Java Database Connectivity
JIT	Just-In-Time Compilation
JLS	Java Language Specification

JNDI	Java Naming and Directory Interface
JVM	Java Virtual Machine
LOV	List of Values
MTS	Multi-Threaded Server
OCI	Oracle Call Interface
OCS	Oracle Consulting Services
ORB	Object Request Broker, a feature of CORBA
OSS	Oracle Support Services
OTM	Oracle Telephony Manager
PL/SQL	Programming Language/Structured Query Language
RAD	Rapid Application Development
RDBMS	Relational Database Management System
RUP	Rolled-Up Patch
SID	System Identifier
SOA	Scope, Objectives and Approach
SQLJ	Structured Query Language (SQL) embedded in Java
SysAdmin	Systems Administrator
TTC	Two-Task Common
UI	User Interface
UWQ	Oracle Universal Work Queue

For More Information

For more information on the acronyms described here—for example, for definitions of many of these terms—see the glossary in Oracle Applications Concepts Release 11*i*.

B

Oracle Scripting Tutorial Exercises

The step-by-step procedures documented in these exercises describe how to create a simple script, how to modify it by adding panels and panel text, how to create panels with lookup values (for multiple answer choices) and distinct branching, how to create a group and populate it with a shortcut that enables the Disconnect button in the Scripting Engine, and how to use two different command types, the ***constant*** command and the ***blackboard*** command. The resulting script does not require custom Java commands and therefore does not depend on loading compiled custom Java classes to the database.

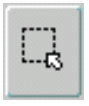
Note that if you are creating a script in order to test an Oracle Scripting implementation, it is recommended you complete, at minimum, exercises one through four, and use the resulting final script as a simple test script with no custom Java methods.

The fifth exercise is included in order to familiarize you with the procedures for creating and executing commands in a script.

For more information on using Oracle Scripting, refer to the *Oracle Scripting Concepts and Procedures* document.

Exercise B-1—Create a Simple Script

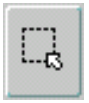
The script you will create by following the instructions for Exercise 1 in the next pages simply has an introductory panel which displays the text “Hello, World.” It then allows the user to press a button labelled Continue, which results in the termination of the script.

Step No.	Required	Step-by-Step Description	Perform From
1.	Required	On the client PC which contains the Oracle Scripting Author, select the **Start** menu.	Windows Desktop
2.	Required	From the Programs listing, highlight **Oracle Scripting**, then select **Scripting Author**. This opens the Scrip Author development environment.	Windows Desktop
3.	Required	From the Oracle Scripting **File** menu, select **New,** or click on the **New Script** icon from the Toolbar. Oracle Scripting creates a new, empty script within the Script Author’s Visual Layout region (the “canvas“). An icon called a **Start** node appears in the canvas, indicating that the new script has been started, and the Script Count in the Author’s Explorer area will be shown as 1.	File menu *or* Script Author Toolbar 
4.	Required	Click the **Panel Insertion** icon from the Toolbar to indicate you wish to insert a new panel.	 Script Author Toolbar
5.	Required	Click ***once*** in the canvas somewhere below the Start node. This action places a panel icon (shown as a blue sphere) into the canvas. **Note:** Until you switch to another Toolbar object or the Toggle Selection Mode (selected in the next step), the current Toolbar object remains selected and will continue to place objects (in this case, panels) on the canvas each time the mouse is clicked in the work area.	Script Author canvas
6.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	Script Author Toolbar 

7.	Required	**Right-click** on the panel that you just added to the script. The Script Author displays a drop down menu with the following options: Edit, Edit Blob Name, and Edit Blob properties. **Note:** If the dialog displayed shows only two options, Edit and Edit Blob properties, then you have not correctly selected the Panel node. Left-click to close the menu and re-select the Panel node.	Script Author canvas
8.	Required	Select the **Edit Blob Properties** option from the menu created by right-clicking on the new panel. The Script Author displays the Panel**[n]** Properties dialog. **Note:** The Script Author automatically assigns numbering to each new panel created (represented by **[n]** above) to enforce unique panel names. When you type a value in the Name field of the panel properties (such as "Hello") and apply the change, the value you entered replaces the **[n]** as the new name of the panel you have just created. Thus, for example, Panel5 Properties becomes PanelHello Properties. For the sake of simplicity, the name of each new panel (the value of **[n]**) will be excluded in these instructions, and the dialog will be referred to generically as the Panel Properties dialog.	Panel Properties dialog
9.	Required	In the **Panel Properties** dialog, change the values of the following fields: Name: **Hello** Comments: *Leave blank* Label: **Hello**	Panel Properties dialog
10.	Required	Click the **Apply** button in the Panel Properties dialog. **Note:** Clicking the **Apply** button saves the values entered or changed (just as if you had clicked the **Ok** button), but leaves the dialog open for additional work.	Panel Properties dialog
11.	Required	Click on the **Answers** property displayed on the left pane of the Panel Properties dialog. The right pane of the dialog changes to display a text area which will list any answers you define for the script. **Note:** For a script to be syntactically correct, every panel must contain at least one answer that will trigger when the script is executed (i.e., that is established as the default for distinct branching).	Panel Properties dialog
12.	Required	Click the **Add** button which is immediately below the Answers text area. The Answer Entry dialog appears.	Panel Properties dialog

13.	Required	Provide the following values for the fields in the Answer Entry dialog: Default for Distinct Branching: ***checkmark*** Name: **Hello_answer** UI Type: **Button** UI Label: *Leave blank* Click **Ok** to close the dialog. The answer that you have defined now appears in the Answers text area in the Panel Properties dialog. It should appear as an entry with the name **Answer:Hello_answer (default)**. **Note:** If the word "default" does not appear in parenthesis next to the answer you have just defined, use the Edit button to re-enter the Answer Entry dialog and click on the Default for Distinct Branching checkbox. Click **Ok** to apply, and then go to the next step.	Answer Entry dialog
14.	Required	From the Answers pane of the Panel Properties dialog, click on the **Edit Data Dictionary** button, which appears just below the list of defined answers for this panel. The application displays the Edit Data Dictionary dialog.	Panel Properties dialog
15.	Required	Select the **Lookups** tab.	Edit Data Dictionary dialog
16.	Required	On the Lookups tab, click on the **Specify Lookups** radio button.	Edit Data Dictionary dialog
17.	Required	Click the **Add** button. The application displays the Lookup Entry dialog.	Lookup Entry dialog
18.	Required	In the Lookup Entry dialog, assign the following values to the fields: Display Value: **Continue** Value: **Continue** Click **Ok** to close the Lookup Entry dialog and return to the Edit Data Dictionary dialog. Click **Ok** to close the Edit Data Dictionary dialog and return to the Panel Properties dialog.	Lookup Entry dialog Edit Data Dictionary dialog
19.	Required	Click **Ok** to close the Panel Properties dialog. The canvas should now show the panel whose name and answer properties you just configured.	Panel Properties dialog

20.	Required	Click *once* on the panel you have just created. When the markers appear around the panel to indicate that you have selected it, open the **Panel Text Editor** (PTE). **Note:** The **Panel Text Editor** can be opened by selecting Panel Text Editor from the **Tools** menu, by clicking on the **Edit Panel Text** icon from the Toolbar, or by entering the keyboard combination command CTRL-SHIFT-E).	Tools menu *or* Script Author canvas *or* keyboard 
21.	Required	In the Panel Text Editor dialog, click on the **Spoken Text Mode** icon. Then move your cursor to the Text entry area and type the following text: **Hello, World!**	 PTE Toolbar
22.	Required	Save the panel text you have created and exit the Panel Text Editor. You can do this by selecting Save and then Exit (both from the **File** menu), or you can simply exit the Panel Text Editor and click on the **Yes** button when the Unsaved Changes dialog asks if you would like to save the changes you have made.	Panel Text Editor *or* Unsaved Changes dialog
23.	Required	The Panel Text Editor closes and the Script Author tool has focus. Select the **Termination Point Insertion Mode** icon from the Toolbar.	 Script Author Toolbar
24.	Required	**Click once** in the canvas somewhere below the single panel in the script. This will place a termination node into the script, which is required for each graph a script uses.	Script Author canvas
25.	Required	From the Script Author Toolbar, select the **Default Branch Mode** icon. The cursor turns into a cross-hairs symbol, indicating that you can begin drawing a line to define flow from one node to another within the canvas.	 Script Author Toolbar
26.	Required	With the cursor displayed as a cross-hairs, use click-and-drag techniques to **draw a line** between the Start and Hello icons, and draw a second line between the Hello icon to the Termination node.	Script Author Visual Layout

27.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	Script Author Toolbar 
28.	*Optional*	Using click-and-drag techniques, select all the objects on the canvas (Start node, Hello panel, Termination node, and Default branches). From the lower Tools menu, select the **Align Selected Objects Vertically** icon; all highlighted items will align vertically, presenting a neat graph on the canvas.	 Script Author Toolbar
29.	Required	At the **File** menu, select **Save** to save your work. This command can also be executed by selecting the Save Script (disk) icon in the Toolbar or by entering CTRL-S from the keyboard.	Script Author File Menu *or* Script Author Toolbar *or* keyboard
30.	Required	In the resulting Script Chooser menu, fill in the following: Location: Select the directory in which you wish to save the file. File Type: Select the default **Oracle CRM script file (.script)** File name: **practice_script_no_Java_1** Click the **Save** button to save the file in the selected directory. **Note:** For location, we suggest **a child directory of the Script Author directory**, for example (your path may differ): `"[OraHome]\oracle\apps\ies\scripts."`	Script Chooser dialog
31.	Required	Even though you have saved and named your script in the Script Author client's filing system, you still need to assign a name to the global script, which will be recognized from a database perspective. Right-click on any unpopulated section of the canvas to display two options, Edit and Edit Blob Properties, and select Edit Blob Properties to open the [Script Name] Properties dialog. You can also open this dialog by selecting **Script Properties** from the **File** menu. **Note:** If the dialog displayed shows *three* options—Edit, Edit Blob Name, and Edit Blob Properties—then you have not correctly selected the Script properties node, but properties for another object. The global script itself is the Binary Large Object (Blob) whose properties you want to modify. In this case, left-click to close the menu and re-select the **Script Properties** following one of the methods above.	Script Author canvas

32.	Required	In the resulting global [Script Name] Properties dialog (which should currently be named something like **Untitled1 Properties**), fill in the following: Name: **practice_script_no_Java_1** Comments: *Leave blank.* Script language: **AMERICAN**, for American English. Click the **Ok** button to save the global script properties and return to the canvas. **Note:** For the Name field, fill in a name under which this script will be deployed to the database. We recommend using the same name as that you assigned to the `.script` file in the filing system, if practical. For the Comments field, you may elect to fill this in with information you will find helpful to identify this particular script or script version. This field is for information only and is not processed in any way. For the Script language field, enter the language in which this script is developed, based on the FND_LANGUAGES able of the Oracle 8*i* database. For this example, ensure this field reads **AMERICAN**, for American English.	Global [Script Name] Properties dialog
33.	Suggested	Select **Syntax Check** from the **Tools** menu, or click on the Check Syntax icon in the Toolbar. Assuming all the syntax in your script is correct, a status line at the bottom of the Author will display the message “Syntax check successful, with no errors.” If the syntax check fails, correct any errors and check the syntax of the script again until successful.	Tools menu or Check Syntax Toolbar icon
34.	Required	At the **File** menu, select **Save** again, since your script now has a name in the Global Script properties. **Note:** The method used to save the document is not important. You may elect to click on the Save icon in the Script Author Toolbar or enter CTRL-S from the keyboard.	File Menu *or* Script Author Toolbar *or* keyboard
35.	Required	From the Tools menu, select **Deploy Script**. This displays the Deploy Script To dialog. Populate the appropriate fields as described in detail in the Using Oracle Scripting > Compiling and Deploying a Script section of *Implementing Scripting R11i (*the Scripting Implementation Guide or IG), or in the Using Oracle Scripting Author > Deploying a Script to the Database section in the *Oracle Scripting Concepts and Procedures Guide* for R11*i*.	Deploy Script To dialog

36.	Required	At this point, your script should look something like the Figure B-1 below.	Script Author Visual Layout
37.	Required	Launch the Scripting Engine by using the prescribed manner for your environment. (For more information, refer to the sections Using Oracle Scripting > How to Launch Oracle Scripting and Using Oracle Scripting > Using the Scripting Engine in the Scripting IG. Select the script created in this exercise, **practice_script_no_Java_1**, from the Script Chooser, and launch the script. One panel will result, and upon clicking the Continue button, the script will terminate. **Note:** Do not select the Disconnect button in the Scripting Engine GUI. This will not be enabled until a subsequent exercise.	Script Engine

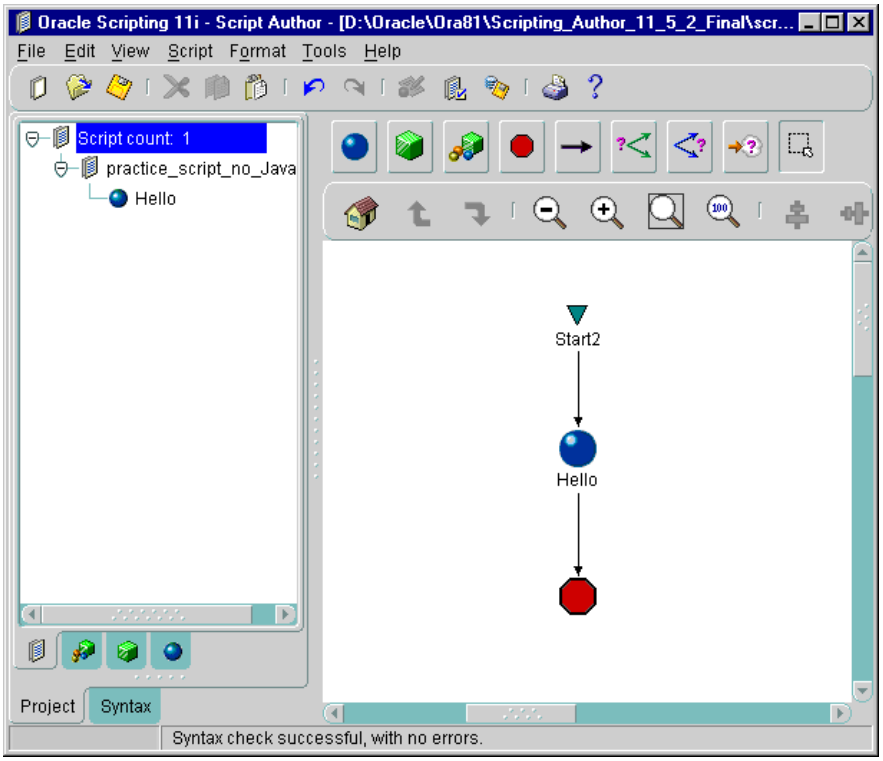


Figure B–1 Figure B-1. Completed practice_script_no_Java_1 script.

Concepts Demonstrated in Exercise B-1

Launching a Script

The last step in Exercise B-1 (and each subsequent exercise) is to launch the completed practice script in the Scripting Engine using the prescribed method for your environment. For more information, refer to the sections Using Oracle Scripting > How to Launch Oracle Scripting and Using Oracle Scripting > Using the Scripting Engine.

Answer Requirements

Each panel must contain at least one answer that is named and triggered to default. Optionally, you may choose to include a UI label that will appear beside the answer control at runtime. For some user interface (UI) types for an answer, no other steps are necessary. For example, if the UI type is a text field or text area, a checkbox, or a single button, an agent must simply enter the answer (by typing or clicking) at runtime. In the case of a checkbox, if left unchecked, the agent must click a Continue button in order to register the unchecked box as the response.

Answers having UI types for which the agent must choose one of several preprogrammed values require other processing in the Script Author to enter these lookup values, as seen in Exercise B-1. These UI types include radio buttons, drop down lists, or button controls that display more than one button.

Key Value Pairs

It is preferable to think of panels containing one or more *question and answer pairs.* Values entered into the Scripting Engine (when an agent provides an answer in a panel, or as the result of a command) are retained automatically, for the duration of the interaction, in a virtual memory space referred to as the "Scripting blackboard." These values are stored in key value pairs consisting of the *combination* of the "question" (identified by the answer name) and the "answer" (the value that was provided in the Scripting Engine, either as a result of the command or due to the answer provided by the agent).

These values are available immediately after that point in the Script for Scripting to use. A value in the blackboard can be displayed in the text of a panel, or in a static display area at the top of the Scripting Engine GUI. The value can be evaluated and

can trigger events based on criteria you specify in custom code associated with a command in the Script Author. A value can also control flow of the script.

If this value was supplied as an answer, it does not matter what the answer UI type displayed in the Scripting Engine is. For example, it is not relevant whether that answer was typed into a text field or text area; or was selected from a preprogrammed List of Values (LOV) entered into scripting as a "Lookup Value" for radio buttons, buttons, or a dropdown list; nor is it relevant if the answer was associated to that question by either checking or leaving unchecked a checkbox in the particular panel.

Naming Convention for Question and Answer Pairs

These exercises use an answer naming convention of **[panelname]_answer**. Based on the preferences of the author of a script, or on predetermined requirements for a specific project, differing naming conventions might be used.

Some authors of scripts name each answer by the type of user interface (UI) for the particular answer. Some authors name each answer by the type of user interface (UI) for the particular answer. Using this convention, the single answer defined in Exercise B-1 would be named **Hello_btn** since this answer will appear as a button.

Thus, based on the available UI types available for an answer, you might see or use conventions similar to the following.

Answer UI Type	Naming Convention	Sample
Text Field	[panelname]_fld	Hello_fld
Text Area	[panelname]_area	Hello_area
Radio Button	[panelname]_rbtn	Hello_rbtn
Checkbox	[panelname]_ckbx	Hello_ckbx
Button	[panelname]_btn	Hello_btn
Drop Down	[panelname]_ddn	Hello_ddn
Password	[panelname]_pswd	Hello_pswd

Multiple Answers in a Panel

While panels must contain at least one question and answer pair, they may contain two or more. There is no practical limit, although the product itself lends towards

keeping answers distinct. The most frequently encountered situation in which a panel will contain multiple question and answer pairs is that of collecting a set of related data points, such as the multiple values required for an address.

If your script naming conventions follow the **[panelname]_answer** model, you will have to revise it slightly to account for uniqueness. In exercise B-1, the Hello panel had a single answer of **Hello_answer**. If it had two question and answer pairs (for example, to collect your name and gender), you may consider following a **[panelname]_[topic]_answer** convention. For this example the first question could be named **Hello_name_answer**, and the second, **Hello_gender_answer**.

Exercise B-2—Modify a Simple Script

The script you created in Exercise 1 will be extended by adding a second panel and reassigning branching for the revised script flow. When executing the script created above, clicking the Continue button in the Hello panel takes you to the default path—the Termination node—which results in the end of the script. In this lesson we will modify the script by adding a second panel that is the default (and only) path from the Hello panel, which will display the text "Goodbye, World." A Finish button at the end of the script (represented as the only Lookup Value associated with the Goodbye panel) terminates script processing.

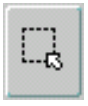
Steps 1 through 4 assume the Script Author is closed. If continuing from Exercise B-1, skip to step 5.

Step No.	Required	Step-by-Step Description	Perform From
1.	Required	On the client PC which contains the Oracle Scripting Author, select the **Start** menu.	Windows Desktop
2.	Required	From the Programs listing, highlight **Oracle Scripting**, then select **Scripting Author**. This opens the Script Author development environment.	Windows Desktop
3.	Required	From the Oracle Scripting **File** menu, select **Open,** or click on the **Open a Script** icon from the Toolbar. From the resulting Script Chooser dialog, navigate to the location on the Script Author client where scripts are stored, and open the script from the previous example, **practice_script_no_Java_1.script.**	File menu *or* Script Author Toolbar 
4.	*Optional*	When an existing script opens, you can change the view of the script by selecting the appropriate view tools in the Toolbar. To view the entire script in the canvas click the Zoom To Fit Graph In Window icon; to see as much of the contents as fit at the standard 100% view, click the Zoom To 100% icon. Doing both in succession orients the view nicely.	  Script Author Toolbar

5.	Required	Rename the script by selecting Save As from the Oracle Scripting **File** menu. In the resulting Script Chooser dialog, rename the script **practice_script_no_Java_2** in the File Name field. **Note:** The Script Chooser will default to the directory last referenced during this session of the Script Author.	File menu, Script Chooser dialog
6.	Required	Change the global script properties to reflect the new name of the script. Right-click on any unpopulated section of the canvas and select Edit Blob Properties, or select **Script Properties** from the **File** menu.	Script Author canvas *or* File menu
7.	Required	According to the global script properties, the script is currently named **practice_script_no_Java_1**. Replace the numeral 1 with 2 in the Name field. No other fields require changes. Name: **practice_script_no_Java_2** Comments: *Leave blank.* Script language: **AMERICAN** Click the **Ok** button to save the global script properties and return to the canvas.	Global [Script Name] Properties dialog
8.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to ensure you can select and move objects on the canvas.	 Script Author Toolbar
9.	Required	Delete the existing default branch between the Hello panel and the Termination node. Click on the default branch until it changes from black to red, indicating that it is selected. Then, delete the branch by pressing the Delete key on your keyboard.	Script Author canvas
10.	Required	Move the Termination node lower in the canvas by clicking on it once to select it, and dragging it lower on the canvas.	Script Author canvas
11.	Required	Click the **Panel Insertion** icon from the Toolbar to indicate you wish to insert a new panel.	 Script Author Toolbar

12.	Required	Click ***once*** in the canvas directly below the Hello panel in the newly vacated space. This action places a new untitled panel into the canvas.	Script Author canvas
13.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	 Script Author Toolbar
14.	Required	**Right-click** on the new panel and select the **Edit Blob Properties** option to display the Panel Properties dialog.	Script Author canvas
15.	Required	In the **Panel Properties** dialog, change the values of the following fields: Name: **Goodbye** Comments: *Leave blank* Label: **Goodbye** **Note:** In the Script Author, when you open a dialog, the focus returns to the last field you visited. in this case, if focus is on the Panel Answer, click on the Properties node in the left pane of the Panel Properties dialog.	Panel Properties dialog
16.	Required	Click the **Apply** button in the Panel Properties dialog.	Panel Properties dialog
17.	Required	Click on the **Answers** property displayed on the left pane of the Panel Properties dialog. The right pane of the dialog changes to display a text area which will list any answers you define for the script.	Panel Properties dialog
18.	Required	Click the **Add** button which is immediately below the Answers text area. The Answer Entry dialog appears.	Panel Properties dialog

19.	Required	Provide the following values for the fields in the Answer Entry dialog: Default for Distinct Branching: ***checkmark*** Name: **Goodbye_answer** UI Type: **Button** UI Label: *Leave blank* Click **Ok** to close the dialog. The answer that you have defined now appears in the Answers text area in the Panel Properties dialog. It should appear as an entry with the name **Answer:Goodbye_answer (default)**. **Note:** If the word "default" does not appear in parenthesis next to the answer you have just defined, use the Edit button to re-enter the Answer Entry dialog and click on the Default for Distinct Branching checkbox. Click **Ok** to apply, and then go to the next step.	Answer Entry dialog
20.	Required	From the Answers pane of the Panel Properties dialog, click on the **Edit Data Dictionary** button. The application displays the Edit Data Dictionary dialog.	Panel Properties dialog
21.	Required	Select the **Lookups** tab.	Edit Data Dictionary dialog
22.	Required	On the Lookups tab, click on the **Specify Lookups** radio button.	Edit Data Dictionary dialog
23.	Required	Click the **Add** button. The application displays the Lookup Entry dialog.	Lookup Entry dialog
24.	Required	In the Lookup Entry dialog, assign the following values to the fields: Display Value: **Finish** Value: **Finish** Click **Ok** to close the Lookup Entry dialog and return to the Edit Data Dictionary dialog. Click **Ok** to close the Edit Data Dictionary dialog and return to the Panel Properties dialog.	Lookup Entry dialog Edit Data Dictionary dialog
25.	Required	Click **Ok** to close the Panel Properties dialog. The canvas should now show the panel whose name and answer properties you just configured.	Panel Properties dialog

26.	Required	Click *once* on the panel you have just created. When the markers appear around the panel to indicate that you have selected it, open the **Panel Text Editor** (PTE). **Note:** The **Panel Text Editor** can be opened by selecting Panel Text Editor from the **Tools** menu, by clicking on the **Edit Panel Text** icon from the Toolbar, or by entering the keyboard combination command CTRL-SHIFT-E).	Tools menu *or* Script Author canvas *or* keyboard 
27.	Required	In the Panel Text Editor dialog, click on the **Spoken Text Mode** icon. Then move your cursor to the Text entry area and type the following text: **Goodbye, World!**	 PTE Toolbar
28.	Required	Save the panel text you have created and exit the Panel Text Editor.	Panel Text Editor *or* Unsaved Changes dialog
29.	Required	The Panel Text Editor closes and the Script Author tool has focus.	Script Author canvas
30.	Required	From the Script Author Toolbar, select the **Default Branch Mode** icon. The cursor turns into a cross-hairs symbol, indicating that you can begin drawing a line to define flow from one node to another within the canvas.	 Script Author Toolbar
31.	Required	With the cursor displayed as a cross-hairs, use click-and-drag techniques to **draw a line** between the Hello and Goodbye icons, and draw a second line between the Goodbye icon to the Termination node.	Script Author Visual Layout
32.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	Script Author Toolbar 
33.	*Optional*	Using click-and-drag techniques, select all the objects on the canvas (Start node, Hello panel, Termination node, and Default branches). From the lower Tools menu, select the **Align Selected Objects Vertically** icon; all highlighted items will align vertically, presenting a neat graph on the canvas.	 Script Author Toolbar

34.	Suggested	Select **Syntax Check** from the **Tools** menu, or click on the Check Syntax icon in the Toolbar. Assuming all the syntax in your script is correct, a status line at the bottom of the Author will display the message "Syntax check successful, with no errors." If the syntax check fails, correct any errors and check the syntax of the script again until successful.	Tools menu *or* Check Syntax Toolbar icon
35.	Required	Save this script again, to ensure all changes you have made since the script was renamed at the beginning of this exercise are retained.	File menu *or* Script Author Toolbar *or* keyboard
36.	Required	From the Tools menu, select **Deploy Script**. Populate the appropriate fields to reflect your applications database location and deploy the script as before.	Deploy Script To dialog
37.	Required	At this point, your script should look something like the Figure B-2 below.	Script Author Visual Layout
38.	Required	Launch the Scripting Engine by using the prescribed manner for your environment, select **practice_script_no_Java_2**, from the Script Chooser, and launch the script. Clicking Continue at the Hello panel should bring you to the Goodbye panel. Upon clicking the Finish button in the Goodbye panel, the script will terminate. **Note:** Do not select the Disconnect button in the Scripting Engine GUI. This will not be enabled until a subsequent exercise.	Script Engine

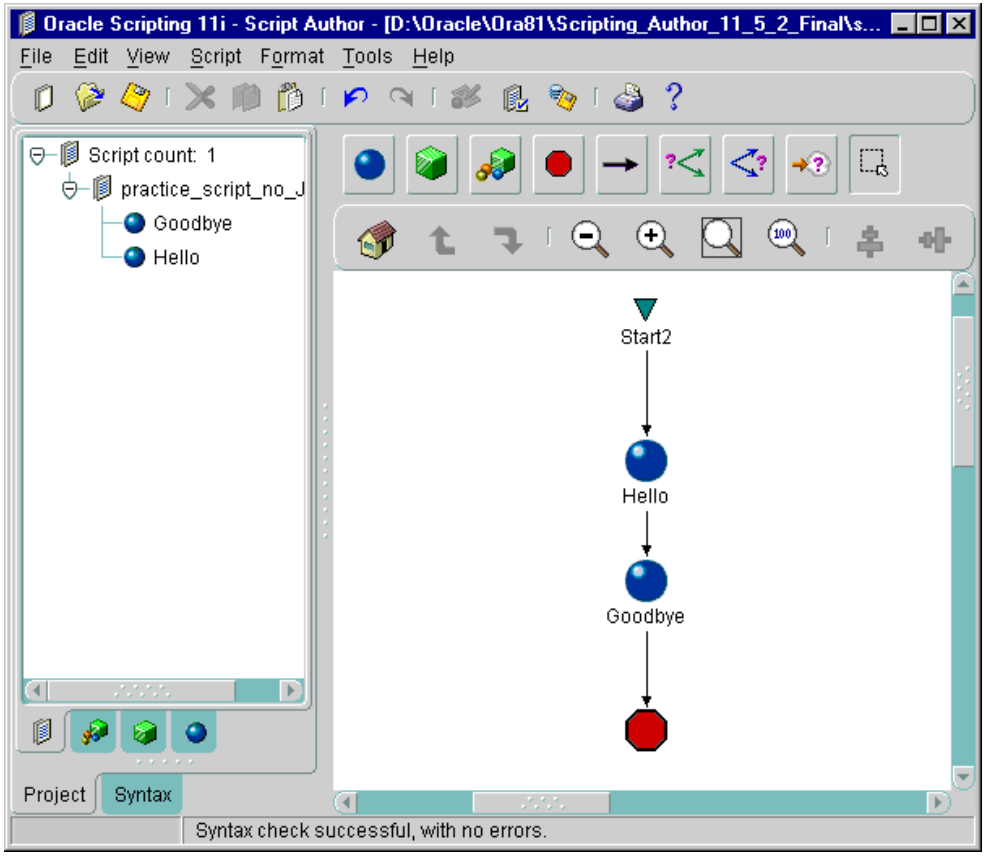


Figure B–2 Figure B-2. Revised script practice_script_no_Java_2 script includes another panel and new default branching to Termination node.

Concepts Demonstrated in Exercise B-2

Modifying or Moving Objects on the Canvas

Sticky Tools

The Script Author uses the “sticky tools“ paradigm for its object insertion tools on the Toolbar. After using any of the insertion mode tools to place Script Author

objects (panels, groups, blocks, branches, or terminal nodes) on the canvas, you will find that you must click on the **Toggle Selection Mode** icon from the Toolbar to deselect that insertion mode before you are able to select, highlight, move, or delete any of the existing objects on the canvas.

If you forget to switch to Toggle Selection mode, clicking on the canvas results in placing on the canvas another of the most recently selected Scripting object. This can be deleted (*after* you switch to Toggle Selection Mode) by highlighting the unwanted object and deleting it using the prescribed key (Delete or Backspace) from your keyboard.

Multiple Methods

As you have seen from the first two exercises, there are multiple methods to accomplish most goals. For example, there are keyboard commands, menu commands and Toolbars to perform many of the same tasks. The Script Author is intended to be a tool suitable for a variety of working styles. No one way is "more correct;" use whichever methods are comfortable to you.

Common Script Modification Pitfalls

One common pitfall is neglecting to change a global script property name when deploying a new script to the database. If you forget this step you will overwrite the previous version of the script with that name. Changing the name in the filing system is not sufficient; the database only recognizes the name of the script from its Name property, accessed by (1) double clicking on any unpopulated section of the canvas, (2) right-clicking on any unpopulated section of the canvas and selecting Edit Blob Properties, or (3) selecting Script Properties from the **File** menu.

When performing rapid edits to a script, another common pitfall is forgetting to add panel text, since the text for a panel is not accessed in the same manner that the panel's other properties are set or modified. This will only be caught when testing a script.

Exercise B-3—Distinct Branching and Multiple Lookup Values

In this exercise you will extend the script from exercise B-2 by adding one panel with multiple possible answer values (referred to as lookup values). Each of the lookup values, when selected at runtime, will lead to a separate flow in the script. This is achieved using distinct branching that you will create below. These three paths are represented by three new panels you will add, as detailed in the following steps. In this practice script, all three divergent paths will again converge to the same path, as represented by the Goodbye panel created in the previous exercise.

Steps 1 through 4 assume the Script Author is closed. If continuing from Exercise B-2, skip to step 5.

Step No.	Required	Step-by-Step Description	Perform From
1.	Required	On the client PC which contains the Oracle Scripting Author, select the **Start** menu.	Windows Desktop
2.	Required	From the Programs listing, highlight **Oracle Scripting**, then select **Scripting Author**. This opens the Script Author development environment.	Windows Desktop
3.	Required	From the Oracle Scripting **File** menu, select **Open,** or click on the **Open a Script** icon from the Toolbar. From the resulting Script Chooser dialog, navigate to the location on the Script Author client where scripts are stored, and open the script from the previous example, **practice_script_no_Java_2.script.**	File menu *or* Script Author Toolbar 
4.	*Optional*	If necessary, click the Zoom To Fit Graph In Window icon and then click the Zoom To 100% icon to orient the scripting graph on the canvas.	  Script Author Toolbar
5.	Required	Rename the script by selecting Save As from the Oracle Scripting **File** menu. In the resulting Script Chooser dialog, rename the script **practice_script_no_Java_3** in the File Name field.	File menu, Script Chooser dialog

6.	Required	Change the global script properties to reflect the new name of the script. Right-click on any unpopulated section of the canvas and select Edit Blob Properties, or select **Script Properties** from the **File** menu.	Script Author canvas *or* File menu

Step No.	Required	Step-by-Step Description	Perform From
7.	Required	According to the global script properties, the script is currently named **practice_script_no_Java_2**. Replace the numeral 2 with 3 in the Name field. No other fields require changes. Name: **practice_script_no_Java_3** Comments: *Leave blank.* Script language: **AMERICAN** Click the **Ok** button to save the global script properties and return to the canvas.	Global [Script Name] Properties dialog
8.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to ensure you can select and move objects on the canvas.	 Script Author Toolbar
9.	Required	Delete the existing default branch between the Hello panel and the Goodbye panel by clicking on the default branch until it changes from black to red, and then pressing the Delete key on your keyboard.	Script Author canvas
10.	Required	Move the Goodbye panel, default branch, and Termination node several inches lower in the canvas by selecting those objects and moving the cursor lower in the graph with all objects selected.	Script Author canvas
11.	Required	Click the **Panel Insertion** icon from the Toolbar to indicate you wish to insert a new panel.	 Script Author Toolbar
12.	Required	Click ***once*** in the canvas directly below the Hello panel in the newly vacated space. This action places a new untitled panel into the canvas.	Script Author canvas

13.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	 Script Author Toolbar
14.	Required	**Right-click** on the new panel and select the **Edit Blob Properties** option to display the Panel Properties dialog.	Script Author canvas
15.	Required	In the **Panel Properties** dialog, change the values of the following fields: Name: **Category** Comments: *Leave blank* Label: **Category**	Panel Properties dialog
16.	Required	Click the **Apply** button in the Panel Properties dialog.	Panel Properties dialog
17.	Required	Click on the **Answers** property displayed on the left pane of the Panel Properties dialog.	Panel Properties dialog
18.	Required	Click the **Add** button to pop the Answer Entry dialog.	Panel Properties dialog
19.	Required	Provide the following values for the fields in the Answer Entry dialog: Default for Distinct Branching: ***checkmark*** Name: **Category_answer** UI Type: **Radio Button** UI Label: *Leave blank* Click **Ok** to close the dialog. The answer that you have defined now appears in the Answers text area in the Panel Properties dialog. It should appear as an entry with the name **Answer:Category_answer (default)**.	Answer Entry dialog
20.	Required	From the Answers pane of the Panel Properties dialog, click on the **Edit Data Dictionary** button. The application displays the Edit Data Dictionary dialog.	Panel Properties dialog
21.	Required	Select the **Lookups** tab.	Edit Data Dictionary dialog
22.	Required	On the Lookups tab, click on the **Specify Lookups** radio button.	Edit Data Dictionary dialog
23.	Required	Click the **Add** button. The application displays the Lookup Entry dialog.	Lookup Entry dialog

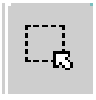
24.	Required	In this first Lookup Entry dialog, assign the following values to the fields: Display Value: **Sales** Value: **Sales** Click **Ok** to close the Lookup Entry dialog and return to the Edit Data Dictionary dialog.	Lookup Entry dialog Edit Data Dictionary dialog
25.	Required	The Edit Data Dictionary field now contains one entry, showing the lookup value's corresponding value field, a colon, and then its display value field contents. In this case it appears as **Sales:Sales**. Click the **Add** button to add a second lookup value for this answer.	Lookup Entry dialog
26.	Required	In this second Lookup Entry dialog, assign the following values to the fields: Display Value: **Customer Service** Value: **Service** Click **Ok** to close the Lookup Entry dialog and return to the Edit Data Dictionary dialog.	Lookup Entry dialog Edit Data Dictionary dialog
27.	Required	The Edit Data Dictionary field now contains two entries. The new entry appears as **Service:Customer Service**. Click the **Add** button to add a third lookup value for this answer.	Lookup Entry dialog
28.	Required	In this third and final Lookup Entry dialog, assign the following values to the fields: Display Value: **Other** Value: **Other** Click **Ok** to close the Lookup Entry dialog and return to the Edit Data Dictionary dialog.	Lookup Entry dialog Edit Data Dictionary dialog
29.	Required	The Edit Data Dictionary field now contains three entries. All of these are now preprogrammed answers to the panel that can be chosen by a mouse click on the corresponding radio button when this script runs in the Scripting Engine. Click **Ok** to close the Edit Data Dictionary dialog and return to the Panel Properties dialog.	Edit Data Dictionary dialog
30.	Required	Click **Ok** to close the Panel Properties dialog. The canvas should now show the panel whose name and answer properties you just configured.	Panel Properties dialog

31.	Required	Click *once* on the panel you have just created. When the markers appear around the panel to indicate that you have selected it, open the **Panel Text Editor** (PTE).	Tools menu *or* Script Author canvas *or* keyboard 
32.	Required	In the Panel Text Editor dialog, click on the **Instruction Text Mode** icon. and type the following instructions to the agent, followed by a hard return: **[Determine customer's reason for calling and select one:]**	 PTE Toolbar
33.	Required	In the Panel Text Editor dialog, click on the **Spoken Text Mode** icon. Then move your cursor to the Text entry area underneath the previous text and type the following text for the agent to read to the customer: **How can I help you today?**	 PTE Toolbar
34.	Required	Save the panel text you have created and exit the Panel Text Editor.	Panel Text Editor *or* Unsaved Changes dialog
35.	Required	From the Script Author Toolbar, select the **Default Branch Mode** icon, and use click-and-drag techniques to **draw a line** between the Hello and Category icons.	 Script Author Toolbar
36.	Required	Click the **Panel Insertion** icon from the Toolbar to indicate you wish to insert a new panel.	 Script Author Toolbar
37.	Required	Three panels will be created in a horizontal line below the Category panel. Click ***once*** in the canvas below and to the left of the Category panel, placing a new untitled panel into the canvas.	Script Author canvas
38.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	 Script Author Toolbar

39.	Required	**Right-click** on the new panel and select the **Edit Blob Properties** option to display the Panel Properties dialog.	Script Author canvas
40.	Required	In the **Panel Properties** dialog, change the values of the following fields: Name: **SalesCategory** Comments: *Leave blank* Label: **SalesCategory**	Panel Properties dialog
41.	Required	Click the **Apply** button in the Panel Properties dialog.	Panel Properties dialog
42.	Required	Click on the **Answers** property displayed on the left pane of the Panel Properties dialog.	Panel Properties dialog
43.	Required	Click the **Add** button to pop the Answer Entry dialog.	Panel Properties dialog
44.	Required	Provide the following values for the fields in the Answer Entry dialog: Default for Distinct Branching: ***checkmark*** Name: **SalesCategory_answer** UI Type: **Button** UI Label: *Leave blank* Click **Ok** to close the dialog. The answer that you have defined now appears in the Answers text area in the Panel Properties dialog. It should appear as an entry with the name **Answer:SalesCategory_answer (default)**.	Answer Entry dialog
45.	Required	From the Answers pane of the Panel Properties dialog, click on the **Edit Data Dictionary** button. The application displays the Edit Data Dictionary dialog.	Panel Properties dialog
46.	Required	Select the **Lookups** tab.	Edit Data Dictionary dialog
47.	Required	On the Lookups tab, click on the **Specify Lookups** radio button.	Edit Data Dictionary dialog
48.	Required	Click the **Add** button. The application displays the Lookup Entry dialog.	Lookup Entry dialog

49.	Required	In the Lookup Entry dialog, assign the following values to the fields: Display Value: **Continue** Value: **Continue** Click **Ok** to close the Lookup Entry dialog and return to the Edit Data Dictionary dialog. Click **Ok** to close the Edit Data Dictionary dialog and return to the Panel Properties dialog. Click **Ok** to close the Panel Properties dialog.	Lookup Entry dialog Edit Data Dictionary dialog Panel Properties dialog
50.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Panel Insertion mode.	 Script Author Toolbar
51.	Required	Click *once* on the panel you have just created. When the markers appear around the panel to indicate that you have selected it, open the **Panel Text Editor** (PTE).	Tools menu *or* Script Author canvas *or* keyboard 
52.	Required	In the Panel Text Editor dialog, click on the **Instruction Text Mode** icon. and type the following text: **[The category of this call is Sales.]**	 PTE Toolbar
53.	Required	Save the panel text you have created and exit the Panel Text Editor.	Panel Text Editor *or* Unsaved Changes dialog
54.	Required	Click on the new **SalesCategory** panel and duplicate it twice, placing each in a horizontal row under the Categories panel. These three panels serve the same function in this practice script, one for each category of caller. **Note:** When a panel is selected, you can duplicate it by selecting Copy and then Paste from the **Edit** menu.	Script Author canvas Edit menu
55.	Required	**Right-click** on the first duplicate panel and select the **Edit Blob Properties** option to display the Panel Properties dialog.	Script Author canvas

56.	Required	In the **Panel Properties** dialog, change the values of the following fields: Name: Change from **SalesCategory** to **ServiceCategory** Comments: *Leave blank* Label: Change from **SalesCategory** to **ServiceCategory** Then click the **Apply** button in the Panel Properties dialog and click on the **Answers** property displayed on the left pane of the Panel Properties dialog.	Panel Properties dialog
57.	Required	Click the **Edit** button to pop the existing Answer Entry dialog and change the answer name from **SalesCategory_answer** to **ServiceCategory_answer**. Leave all other values the same. Click **Ok** to close the Answer Entry dialog. Click **Ok** to close the Panel Properties dialog.	Answer Entry dialog Panel Properties dialog
58.	Required	Click *once* on the Service panel. When the markers appear around the panel to indicate that you have selected it, open the **Panel Text Editor** (PTE).	Tools menu *or* Script Author canvas *or* keyboard 
59.	Required	In the Panel Text Editor dialog, modify the **Instruction Text** by changing the word **Sales** to **Service**.	 PTE Toolbar
60.	Required	Save the panel text you have just modified and exit the Panel Text Editor.	Panel Text Editor *or* Unsaved Changes dialog
61.	Required	**Right-click** on the second of two duplicated Sales panels and select the **Edit Blob Properties** option to display the Panel Properties dialog. Modify the panel properties Name and Label, replacing each occurrence of the word “Sales“ with “Other.“ Then modify the answer name in the Answer Entry dialog from SalesCategory_answer to OtherCategory_answer.	Script Author canvas Panel Properties dialog Answer Entry dialog
62.	Required	Click *once* on the **Other** panel. When the markers appear around the panel to indicate that you have selected it, open the **Panel Text Editor** (PTE).	Tools menu *or* Script Author canvas *or* keyboard 

63.	Required	In the Panel Text Editor dialog, modify the **Instruction Text** by changing the word **Sales** to **Other**.	 PTE Toolbar
64.	Required	Save the panel text you have just modified and exit the Panel Text Editor.	Panel Text Editor *or* Unsaved Changes dialog
65.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to ensure you can select and move objects on the canvas.	 Script Author Toolbar
66.	*Optional*	Using click-and-drag techniques, select all three category panels. From the lower Tools menu, select the **Align Selected Objects Horizontally** icon; all highlighted items will align horizontally. Align the ServiceCategory panel under the Categories panel.	 Script Author Toolbar
67.	Required	From the Script Author Toolbar, select the **Distinct Branch Mode** icon. The cursor turns into a cross-hairs symbol, indicating that you can begin drawing a line to define flow between two specified panels within the canvas based on the answer provided at runtime.	 Script Author Toolbar
68.	Required	With the cursor displayed as a cross-hairs, use click-and-drag techniques to **draw a line** between the Category and SalesCategory icons. The new distinct branch line is colored red on the canvas, indicating that it is selected.	Script Author canvas
69.	Required	From the Script Author Toolbar, select the **Toggle Select Mode** icon.	 Script Author Toolbar
70.	Required	**Right-click** on the Distinct branch that you just added to the script, making sure you touch the line instead of the text label. The Script Author displays a drop down menu with the following options: Edit, and Edit Branch Properties. **Note:** If the dialog displayed shows only the Edit option, then you have not correctly selected the Branch node. Left-click to close the menu and re-select the Branch node.	Script Author canvas

71.	Required	Select **Edit Branch Properties** from the drop down menu to open the Distinct**[n]** Properties dialog. **Note:** Similar to panels, each Distinct branch is automatically numbered upon creation. For the sake of simplicity, the **[n]** will be excluded from these instructions.	Distinct Branch Properties dialog
72.	Required	The Distinct Branch Properties dialog has three properties displayed on the left pane of the dialog. When the **Properties** value is selected, change the values of the following fields: Name: **Sales** Comments: *Leave blank* Then click the **Apply** button in the Distinct Branch Properties dialog.	Distinct Branch Properties dialog
73.	Required	Select the **Value** property in the Distinct Branch Properties dialog. In the resulting Value pane on the right, two drop down lists appear. The list on the left displays all lookup values that have not yet been assigned to distinct branches. The right menu will list the answer selected for this particular distinct branch. Click on the first item on the left-hand menu, **Sales:Sales**, so that it highlights. The > > button enables.	Distinct Branch Properties dialog
74.	Required	Click on the **> >** button to move the first lookup value to the Selected answer menu. Click **Ok** to close the Distinct Branch Properties dialog. The Distinct branch that you created from the Categories panel to the SalesCategory panel is labeled Sales and will cause the flow of the script to route to the SalesCategory panel when Sales is selected as the appropriate category at runtime.	Distinct Branch Properties dialog
75.	Required	From the Script Author Toolbar, select the **Distinct Branch Mode** icon. The cursor turns into a cross-hairs symbol, indicating that you can begin drawing a line to define flow between two specified panels within the canvas based on the answer provided at runtime.	 Script Author Toolbar
76.	Required	With the cursor displayed as a cross-hairs, use click-and-drag techniques to **draw a line** between the Category and ServiceCategory icons, and another line between the Category and OtherCategory icons.	Script Author canvas

77.	Required	As described above, associate the appropriate lookup values for the two new Distinct branches you have created.	 Script Author Toolbar
78.	Required	From the Script Author Toolbar, select the **Default Branch Mode** icon, and use click-and-drag techniques to **draw a line** from each of the three Category icons to the Goodbye icon.	 Script Author Toolbar
79.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to ensure you can select and move objects on the canvas.	 Script Author Toolbar
80.	*Optional*	Using click-and-drag techniques, select objects on a vertical axis and then click the **Align Selected Objects Vertically** icon from the lower Tools menu. Then select objects on a horizontal axis and click the **Align Selected Objects Horizontally** icon from the same menu. The objects on the canvas should align neatly as you directed.	  Script Author Toolbar
81.	Suggested	Select **Syntax Check** from the **Tools** menu, or click on the Check Syntax icon in the Toolbar. Assuming all the syntax in your script is correct, a status line at the bottom of the Author will display the message “Syntax check successful, with no errors.” If the syntax check fails, correct any errors and check the syntax of the script again until successful.	Tools menu or Check Syntax Toolbar icon
82.	Required	Save this script again, to ensure all changes you have made since the script was renamed at the beginning of this exercise are retained.	File menu *or* Script Author Toolbar *or* keyboard
83.	Required	From the Tools menu, select **Deploy Script**. Populate the appropriate fields to reflect your applications database location and deploy the script as before.	Deploy Script To dialog
84.	Required	At this point, your script should look something like the Figure B-3 below.	Script Author Visual Layout

85.	Required	Launch the Scripting Engine by using the prescribed manner for your environment, select **practice_script_no_Java_3**, from the Script Chooser, and launch the script. Try each of the possible flows in the script by running through the script three times, choosing a different caller category each time. **Note:** Do not select the Disconnect button in the Scripting Engine GUI. This will not be enabled until a subsequent exercise.	Script Engine

Figure B–3

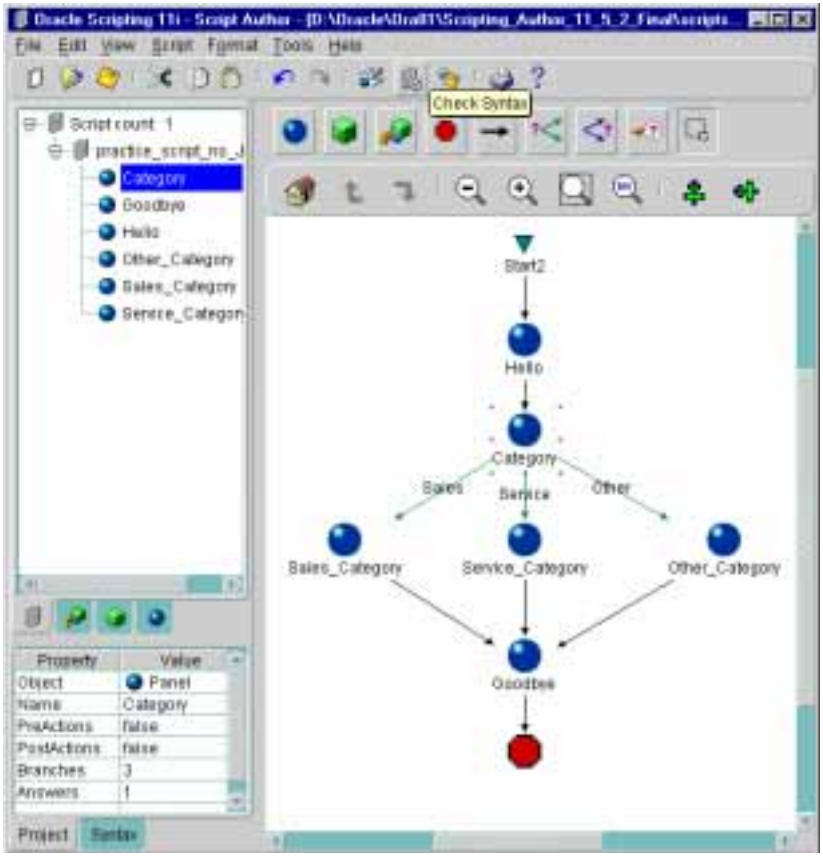


Figure B–4 Figure B-3. Revised script practice_script_no_Java_3 script includes distinct branching from a panel with multiple lookup values for its answer.

Concepts Demonstrated in Exercise B-3

Selecting or Moving Multiple Objects on the Canvas

To select multiple objects on the canvas, you must be in Toggle Selection Mode. This is always accomplished by clicking on the **Toggle Selection Mode** icon from the Toolbar.

There are two ways to select multiple objects on the canvas. First, using click-and-drag techniques, drag the mouse down in a rectangle that encompasses all relevant

items. Alternatively, you can hold down the Shift key and click directly on those multiple objects on the canvas you wish to select.

How to Know an Object Is Selected

You can discern that branches in a graph on a canvas are explicitly selected because each branch type changes from its standard color (black for Default branches, green for Distinct branches, blue for Conditional branches, and orange for Indeterminate branches) to red. Other objects (panels, groups, blocks, and Start and Termination nodes) display selection markers (evenly spaced dots) around the object when selected.

Selecting Associated Branching Automatically

Note that if you wish to move objects that are already attached with the appropriate branch type on the canvas, you need not select the branches.

If, for example, you use Shift-Click techniques to select multiple connected objects such as panels, groups, or blocks (but not the branches connecting them), and then *delete them* from the canvas, the (unselected) branches connected to those objects will also be deleted.

If you select multiple objects (but not their branches) in order to *move them* to another location on the canvas, the branches will stretch or extend from their original positions, remaining attached to and associated with the moved objects in their new locations.

When you explicitly select a branch, it changes from its original color (determined by the type of branch) to *red*, indicating that the branch is selected. However, as in the two examples above, branches that you have *not* explicitly selected (that are attached to objects you *have* selected) will retain their original color on the canvas, rather than changing to red.

Exercise B-4—Creating the WrapUpShortcut Group

This exercise includes the concept of a **group**, which is essentially a subgraph—another canvas in the script. Groups can contain all types of Script Author objects: panels, blocks, all branch types, as well as other groups, with no practical limit. Groups help an author of a script to isolate functionality, making a script more modular.

Groups also may contain a property called a **Shortcut**. A Shortcut enables an agent progressing through a script to jump to another designated location in the script. While this *typically* requires custom Java code, the Script Author has been designed specifically to recognize any group whose shortcut is labelled `WrapUpShortcut`. Any group so labelled will become the destination Shortcut of the Disconnect button, *without requiring custom Java code.* In other words, regardless of where an agent is in the flow of a script, they may click the Disconnect button and will jump to the group designated as the WrapUp group by its Shortcut property.

The WrapUpShortcut group may or may not contain additional panels, and is typically attached to a Termination node by a Default branch. In this way, an agent can end a script prematurely—for example, in the event a customer hangs up the phone prior to the natural end of a script's flow. In this example, we will move the Goodbye panel to become a child panel of the WrapUpShortcut group.

Steps 1 through 4 assume the Script Author is closed. If continuing from Exercise B-3, skip to step 5.

Step No.	Required	Step-by-Step Description	Perform From
1.	Required	On the client PC which contains the Oracle Scripting Author, select the **Start** menu.	Windows Desktop
2.	Required	From the Programs listing, highlight **Oracle Scripting**, then select **Scripting Author**. This opens the Script Author development environment.	Windows Desktop
3.	Required	From the Oracle Scripting **File** menu, select **Open,** or click on the **Open a Script** icon from the Toolbar. From the resulting Script Chooser dialog, navigate to the location on the Script Author client where scripts are stored, and open the script from the previous example, **practice_script_no_Java_ 3.script.**	File menu *or* Script Author Toolbar 

Step No.	Required	Step-by-Step Description	Perform From
4.	*Optional*	If necessary, click the Zoom To Fit Graph In Window icon and then click the Zoom To 100% icon to orient the scripting graph on the canvas.	  Script Author Toolbar
5.	Required	Rename the script by selecting Save As from the Oracle Scripting **File** menu. In the resulting Script Chooser dialog, rename the script **practice_script_no_Java_4** in the File Name field.	File menu, Script Chooser dialog
6.	Required	Change the global script properties to reflect the new name of the script. Right-click on any unpopulated section of the canvas and select Edit Blob Properties, or select **Script Properties** from the **File** menu.	Script Author canvas *or* File menu
7.	Required	According to the global script properties, the script is currently named **practice_script_no_Java_3**. Replace the numeral 3 with 4 in the Name field. No other fields require changes. Name: **practice_script_no_Java_4** Comments: *Leave blank.* Script language: **AMERICAN** Click the **Ok** button to save the global script properties and return to the canvas.	Global [Script Name] Properties dialog
8.	Required	Select the **Group Insertion** icon from the Toolbar.	 Script Author Toolbar
9.	Required	Click **once** towards the bottom of the canvas, close to the Termination node. The Script Author will place a Group node into the script at the insertion point.	Script Author canvas

10.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to turn off the Group Insertion mode.	 Script Author Toolbar
11.	Required	**Right-click** on the newly-added group. The Script Author will display a menu containing the following options: Edit, Edit Blob Name, Edit Blob Properties, and Show Subgraph. Select **Edit Blob Properties** to see the Group Properties dialog. **Note:** If the dialog displayed shows only the options Edit and Edit Blob properties, then you have not correctly selected the Group node. Left-click to close the menu and re-select the Group node.	Script Author canvas
12.	Required	Fill out the fields in the Group Properties dialog in the following manner: Name: **WrapUpGroup** Comment: **Initiates an exit from script** **Note:** The comment is not required.	Group Properties dialog
13.	Required	Click the **Apply** button in the Group Properties dialog.	Group Properties dialog
14.	Required	Click on the **Shortcut** property from the items on the left pane of the Group Properties dialog. The right pane is re-written and a new field appears. You must fill it out as follows: Shortcut: **WrapUpShortcut** Note that the Shortcut name must appear ***exactly as typed here, with upper and lowercase text and no spaces.*** During runtime, when the agent is using the compiled script, the Scripting Engine associates the Disconnect button with a group with the name WrapUpShortcut.	Group Properties dialog
15.	Required	Click the **OK** button in the Group Properties dialog. The dialog closes and the new group appears on the canvas.	Group Properties dialog
16.	Required	Now that there is a new group representing a second graph, we will refer to the main graph that we have worked with for four exercises as the **Root Graph**. From the Script Author Toolbar, select the **Default Branch Node** icon. The cursor turns into a cross-hairs symbol, indicating that you can begin drawing a line to define flow from one node to another within the canvas.	 Script Author Toolbar

17.	Required	With the cursor displayed as a cross-hairs symbol, click and drag to **draw an arrow** between the WrapUpGroup icon and the Termination node on the root graph.	Script Author canvas
18.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar.	 Script Author Toolbar
19.	Required	**Click once** on the Goodbye panel in the root graph. The selection markers will appear.	Script Author canvas
20.	Required	Copy the Goodbye panel. **Note:** You can copy a selected object on the canvas by selecting Copy from the **Edit** menu, using the keyboard command CTRL-C, or by right-clicking on the selected object and choosing Copy from the first of three resulting menus, the **Edit** menu.	Script Author canvas *or* Edit menu *or* keyboard
21.	Required	**Click once** on the WrapUpGroup icon to select it. The selection markers will appear.	Script Author canvas
22.	Required	Click on the **Go Down Into Child Graph** icon from the Toolbar. You will now see a new subgraph, containing only a Start node, as always.	 Script Author Toolbar
23.	Required	Paste the Goodbye panel into the WrapUpGroup subgraph. **Note:** You can paste an object in the clipboard by selecting Paste from the **Edit** menu, using the keyboard command CTRL-V, or by right-clicking on the empty canvas and choosing Paste from the first of two resulting menus, the **Edit** menu.	Script Author canvas *or* Edit menu *or* keyboard
24.	Required	Select the **Termination Point Insertion Mode** icon from the Toolbar.	 Script Author Toolbar
25.	Required	**Click once** in the canvas somewhere below the Goodbye panel icon. This will place a termination node into the script, which is required for each graph a script uses.	Script Author canvas

26.	Required	From the Script Author Toolbar, select the **Default Branch Node** icon. The cursor turns into a cross-hairs symbol, indicating that you can begin drawing a line to define flow from one node to another within the canvas.	 Script Author Toolbar
27.	Required	With the cursor displayed as a cross-hairs, use click-and-drag techniques to **draw a line** between the Start node and the Goodbye panel icon. **Draw a second line** between the Goodbye panel icon and the Termination node. The child graph—the subgraph that contains the contents of the WrapUpGroup—is now syntactically correct.	Script Author canvas
28.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to ensure you can select and move objects on the canvas.	 Script Author Toolbar
29.	*Optional*	Using click-and-drag techniques, select objects on a vertical axis and then click the **Align Selected Objects Vertically** icon from the lower Tools menu. Then select objects on a horizontal axis and click the **Align Selected Objects Horizontally** icon from the same menu. The objects on the canvas should align neatly as you directed.	  Script Author Toolbar
30.	Required	Click on the **Go Up To Parent Graph** icon or the **Go to Root Graph** icon (both on the Toolbar) to return to the previous graph. This script has one parent (the Root Graph) and one child (the WrapUpGroup graph). Groups can contain multiple levels of nested groups with no practical limit.	 Script Author Toolbar
31.	Required	In the root graph, **click on the Goodbye panel** that you previously copied and pasted into the WrapUpGroup. When the selection markers appear, **remove the Goodbye panel** by pressing the Delete key on your keyboard.	Script Author canvas

32.	Required	The default branches from the three specific category panels remain, with a gap where they formerly connected to the Goodbye panel. Move the WrapUpGroup icon to the place previously taken by the deleted panel and reattach default branching to this group. **Note:** There are two ways to restore appropriate branching. (1) You can delete the three dangling default branches, select the **Default Branch Node** icon, and redraw lines from the panels above to the group. (2) You can reattach the dangling branches to the new group object. To do this, move the new WrapUpGroup icon to one of the three dangling default branches until the black branch changes to red. The default branch now associates with the group. Repeat this step twice more for the other dangling branches.	Script Author canvas
33.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar to ensure you can select and move objects on the canvas.	 Script Author Toolbar
34.	*Optional*	Using click-and-drag techniques, select objects on a vertical axis and then click the **Align Selected Objects Vertically** icon from the lower Tools menu. Then select objects on a horizontal axis and click the **Align Selected Objects Horizontally** icon from the same menu. The objects on the canvas should align neatly as you directed.	  Script Author Toolbar
35.	Suggested	Select **Syntax Check** from the **Tools** menu, or click on the Check Syntax icon in the Toolbar. Assuming all the syntax in your script is correct, a status line at the bottom of the Author will display the message “Syntax check successful, with no errors.” If the syntax check fails, correct any errors and check the syntax of the script again until successful.	Tools menu or Check Syntax Toolbar icon
36.	Required	Save this script again, to ensure all changes you have made since the script was renamed at the beginning of this exercise are retained.	File menu *or* Script Author Toolbar *or* keyboard
37.	Required	From the Tools menu, select **Deploy Script**. Populate the appropriate fields to reflect your applications database location and deploy the script as before.	Deploy Script To dialog
38.	Required	At this point, your script should look something like the Figure B-4 below.	Script Author canvas

39.	Required	Launch the Scripting Engine by using the prescribed manner for your environment, select **practice_script_no_Java_4**, from the Script Chooser, and launch the script. As before, try different flows in the script, and also try clicking on the Disconnect button before reaching the natural end of the script flow, to ensure the WrapUpShortcut group is functioning as the destination group.	Script Engine

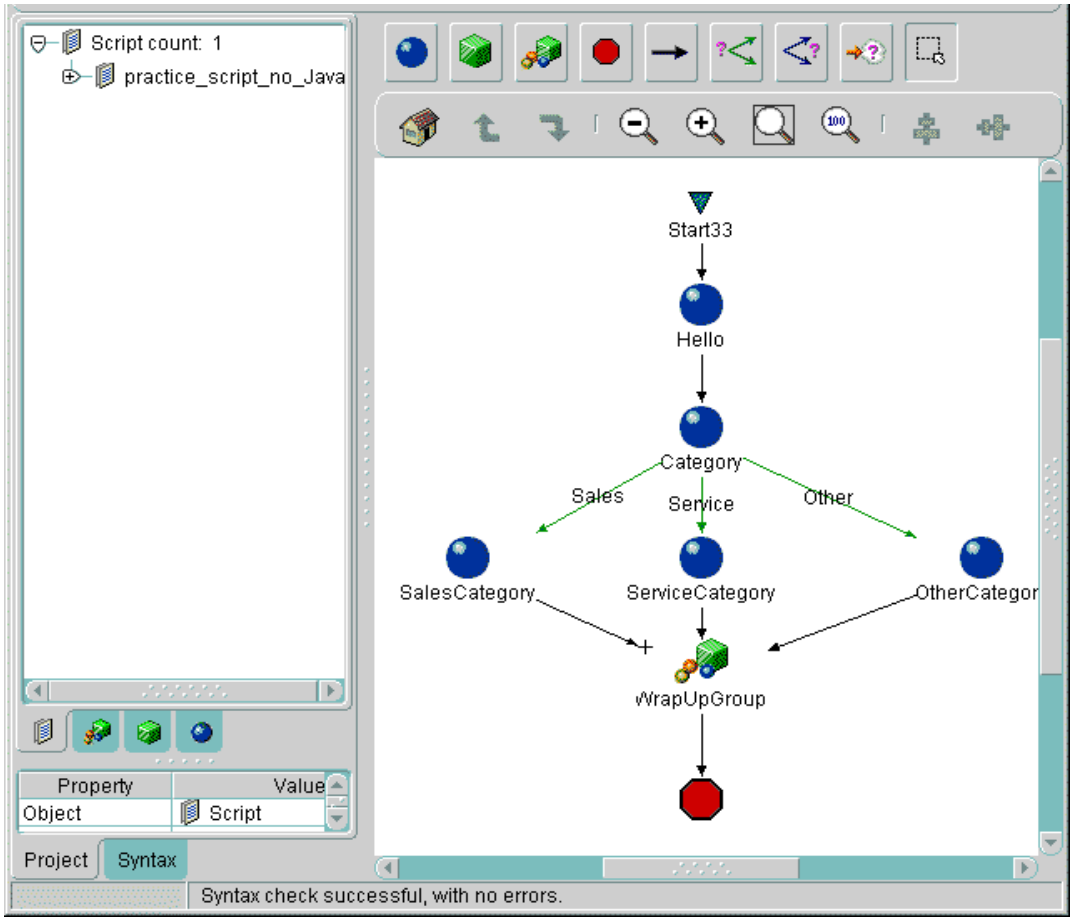


Figure B–5 Figure B-4. Revised script practice_script_no_Java_4 includes WrapUpGroup that contains WrapUpShortcut to enable Disconnect button.

Concepts Demonstrated in Exercise B-4

Requirements for a WrapUpShortcut Group

As shown in Exercise B-4, in order to designate a group as the WrapUpShortcut group, you must modify its Shortcut property in the Group Properties dialog to contain the exact name `WrapUpShortcut`. Doing so designates the group as the destination upon an agent clicking the Disconnect button. Note that the Name property of the group i*s not in any way relevant t*o its functionality as the WrapUpShortcut group. While this script uses the group name **WrapUpGroup**, it could just as easily be called **SayonaraGroup** or **Group24** or **Exit**.

In Exercise B-4, the WrapUpShortcut group contains a panel. Some enterprises may have a requirement for agents to collect from the caller (or enter on their own) information about each call. In these instances, the WrapUpShortcut group acts just like any other group—as a container for other Scripting objects (panels, blocks, other groups) where answers are entered by the agent at runtime in the Scripting Engine GUI.

However, the WrapUpShortcut group may not be required to have any panels or other objects. An enterprise may only require that a script terminate immediately upon the agent clicking the Disconnect button. In this scenario, the WrapUpShortcut group will only contain a Termination node and default branching linking the Start node and the Termination node.

This is because the only other requirement for the WrapUpShortcut group (besides the Shortcut name) is that it follow the same syntactic rules of any group or graph in a script. When a group is created—just like when a new script is created—it contains a Start node. Every graph in every script must be properly terminated. This typically means that every object is connected with appropriate branching and each graph has a Termination node.

Note: Technically, a graph does not require a Termination node *if* it contains an Indeterminate branch. Such a script will pass a syntax check and can be deployed to the database, but will not execute properly unless the branch contains an action referencing an appropriate custom Java command and that custom Java class is deployed to the JServer in the database.

Regardless of whether a group has objects inside or just contains a Start and Termination node connected by a Default branch, when that group is also intended to be the last destination in a script, the group (in its parent graph) is attached by a Default branch to a Termination node.

Copying, Pasting, Deleting and Reattaching Objects on the Canvas

As shown in Exercise B-4, objects can be copied and pasted, allowing you to duplicate functionality of panels, groups, or blocks without needing to perform all steps. Whenever you do this, it is recommended to review the properties of the cloned object.

Panel Properties

Ensure you change the name in the Name and Label field in the Panel Properties section of the Panel Properties dialog. While the Script Author *will* identify a duplicated panel name upon checking the syntax by displaying an error message (**`error: duplicate panel name`** `<name of duplicated panel>`), it will *not* catch a duplicate label name, which could be confusing to an agent using the resulting script.

Answer Properties

Also be sure to change the name in the Answer Properties section of the Answer Entry dialog. This step is crucial to enforce uniqueness of key value pairs (the answer name and the value provided to the panel at runtime).

Cutting, Copying, Pasting, and Deleting Branches

When you *delete* objects on the canvas, the outgoing branches associated with that object typically are deleted as well. (Any branches coming into that object remain on the canvas; these are referred to as *dangling* edges.) When you *cut* objects on the

canvas, the outgoing branches associated with that object are typically removed from the canvas as well, but branches are not retained in the clipboard (for a *cut* or a *copy* operation) unless they were explicitly selected.

Reassociating Branches

In the example above, the Goodbye panel was copied and placed as a child to the WrapUpGroup. Then the original Goodbye panel on the root graph was deleted, leaving three dangling Default branches, and was replaced with the WrapUpGroup. In this example, there are two ways to restore appropriate branching:

1. You can delete the three dangling Default branches, select the **Default Branch Node** icon, and redraw lines from the panels above to the group.
2. You can reattach the dangling branches to the new group object. To do this, move the new WrapUpGroup icon to one of the three dangling Default branches until the black branch changes to red. The Default branch is now associated with the group. Repeat this step for each of the remaining dangling branches.

Clearing Cast Exceptions

Occasionally, when performing an operation (typically one that involves cutting and pasting of text into fields within a dialog), a cast exception will occur, exhibiting an error message in the lower portion of the canvas in a pop-up Error Region. The error may appear similar to the following:

```
Exception occurred during event dispatching:
java.lang.ClassCastException
```

This is an untrapped Java error that is *not* indicating an error on your part. Simply clear the message by right-clicking in the Error Region, and select either menu option, Hide or Clear.

You can confirm there are no serious errors at any time by performing a Syntax Check.

Exercise B-5—Using Commands Where Custom Java Is Not Required

A script can be extended through the use of Commands. Of the Command types available—Java, PL/SQL, Blackboard, Forms, Constant, and COM, only Blackboard and Constant commands can be created without using specific SQL, Java or Forms code. This exercise illustrates slightly more complex functionality of Oracle Scripting by adding both a Blackboard and a Constant command to our script.

Note: COM commands are no longer supported by Oracle Corporation, although the Commands dialog serves as an API to use COM if your situation requires.

Steps 1 through 4 assume the Script Author is closed. If continuing from Exercise B-4, skip to step 5.

Step No.	Required	Step-by-Step Description	Perform From
1.	Required	On the client PC which contains the Oracle Scripting Author, select the **Start** menu.	Windows Desktop
2.	Required	From the Programs listing, highlight **Oracle Scripting**, then select **Scripting Author**. This opens the Script Author development environment.	Windows Desktop
3.	Required	From the Oracle Scripting **File** menu, select **Open,** or click on the **Open a Script** icon from the Toolbar. From the resulting Script Chooser dialog, navigate to the location on the Script Author client where scripts are stored, and open the script from the previous example, **practice_script_no_Java_4.script.**	File menu *or* Script Author Toolbar 

4.	*Optional*	If necessary, click the Zoom To Fit Graph In Window icon and then click the Zoom To 100% icon to orient the scripting graph on the canvas.	  Script Author Toolbar
5.	Required	Rename the script by selecting Save As from the Oracle Scripting **File** menu. In the resulting Script Chooser dialog, rename the script **practice_script_no_Java_5** in the File Name field.	File menu, Script Chooser dialog
6.	Required	Change the global script properties to reflect the new name of the script. Right-click on any unpopulated section of the canvas and select Edit Blob Properties, or select **Script Properties** from the **File** menu.	Script Author canvas *or* File menu
7.	Required	According to the global script properties, the script is currently named **practice_script_no_Java_4**. Replace the numeral 4 with 5 in the Name field. No other fields require changes. Name: **practice_script_no_Java_5** Comments: *Leave blank.* Script language: **AMERICAN** Click the **Ok** button to save the global script properties and return to the canvas.	Global [Script Name] Properties dialog
8.	Required	Click on the **Toggle Selection Mode** icon from the Toolbar.	 Script Author Toolbar
9.	Required	Click *once* on the **Hello** panel.	Script Author canvas
10.	Required	When the selection markers appear, open the **Panel Text Editor** (PTE).	Tools menu *or* Script Author canvas *or* keyboard 

11.	Required	In the PTE dialog, highlight the text **Hello, World**, and replace it with the following text: **May I have your name please?**	 PTE Toolbar
12.	Required	**Save the panel text** you have revised and exit the PTE.	Panel Text Editor *or* Unsaved Changes dialog
13.	Required	**Open the Panel Properties of the Hello panel** by double-clicking on the panel or by right-clicking and selecting Edit Blob Properties.	Script Author canvas
14.	Required	Click on the **Answers** property displayed on the left pane of the Panel Properties dialog. The right pane of the dialog changes to display defined answers for this panel. Highlight the one answer created for this panel, **Answer:Hello_answer (default).**	Panel Properties dialog
15.	Required	**Click once** on the Edit button to edit this answer.	Panel Properties dialog
16.	Required	In the **Answer Entry dialog**, change the values of the following fields: Default for Distinct Branching: *Leave checked.* UI Type: **Text Field** UI Label: **First and Last Name:** **Click Ok** to close the Answer Entry dialog.	Answer Entry dialog
17.	Required	**Click Ok** to close the Panel Properties dialog.	Panel Properties dialog
18.	Required	**Click once** on the WrapUpGroup icon to select it. The selection markers will appear.	Script Author canvas
19.	Required	Click on the **Go Down Into Child Graph** icon from the Toolbar. You will now see the child graph, containing the Start node, Goodbye panel, and the Termination node.	 Script Author Toolbar
20.	Required	Click *once* on the Goodbye panel.	Script Author canvas

21.	Required	When the selection markers appear, open the **Panel Text Editor** (PTE).	Tools menu *or* Script Author canvas *or* keyboard 
22.	Required	The text in the PTE dialog reads **Goodbye, World**. Highlight the word World and click on the **Insert Embedded Value button** in the lower PTE Toolbar.	 PTE Toolbar
23.	Required	Click on the **Modify Properties button** in the upper PTE Toolbar to open the **Command Type dialog.**	 PTE Toolbar
24.	Required	In the **Command Type dialog**, enter the following values: Command Type: **Blackboard Command** Name: **Hello_answer** Command: **Hello_answer** **Note:** The Name and Command values are set to the answer name of the panel whose answer you want to display at runtime in this panel text.	 Command Type dialog
25.	Required	**Click the Add button** in the **Parameters** section of the Command Type dialog to create a new parameter.	 Command Type dialog
26.	Required	In the **Parameters dialog,** enter the following values: Class: *Leave blank.* Name: **Hello_answer** Value: *Leave blank.* In / Out Type: *Unavailable. Leave blank.* Add Value as Command: *Leave unchecked.* **Click Ok** to exit the Parameters dialog and return to the Command Type dialog.	Parameters dialog
27.	Required	**Click Ok** to exit the Command Type dialog.	 Command Type dialog

28.	Required	**Click once** on a blank portion of the PTE dialog. You will see the name of the command you entered into the Command Type dialog, highlighted in yellow to indicate it has an embedded value.	PTE dialog
29.	Required	**Save the panel text** you have revised and exit the PTE.	Panel Text Editor *or* Unsaved Changes dialog
30.	Required	Open the global script properties to add a Constant value so that a company name appears on the Static Information Panel of the script for every session of the script. Right-click on any unpopulated section of the canvas and select Edit Blob Properties, or select **Script Properties** from the **File** menu.	Script Author canvas *or* File menu
31.	Required	Click on the **Static Info Panel** property displayed on the left pane of the global Script Properties dialog. The right pane of the dialog changes to display a three-by-three matrix of areas where information can be displayed on the Static Information Panel during runtime.	Global Script Properties dialog
32.	Required	Click on the top left position in the **Static Info Panel**. Two buttons are enabled on the right of this panel; click **Text**.	Global Script Properties dialog
33.	Required	Three data fields enable below. Populate the fields with information as indicated below: ID: **CompanyName** Label: **Company Name:** (include colon; this will appear as the label on the Static Information Panel.) Click on the **Edit** button to create a Command.	Global Script Properties dialog
34.	Required	In the **Command Type dialog**, enter the following values: Command Type: **Constant Command** Name: **CompanyName** Command: **CompanyName**	 Command Type dialog
35.	Required	**Click the Edit button** in the **Return Value** section of the Command Type dialog to create the Constant parameter to populate in the Static Information Panel.	 Command Type dialog

36.	Required	In the **Parameters dialog,** enter the following values: Class: *Leave blank.* Name: **CompanyName** Value: **Vision Interaction Centers Inc.** In / Out Type: *Unavailable. Leave blank.* Add Value as Command: *Leave unchecked.* **Click Ok** to exit the Parameters dialog and return to the Command Type dialog.	Parameters dialog
37.	Required	**Click Ok** to exit the Command Type dialog.	 Command Type dialog
38.	Required	**Click Apply** to save changes to the Static Information Panel. **Note:** If you were to add more fields into the Static Information Panel, this would be an appropriate time. Click additional fields and define parameters at this time.	Panel Properties dialog
39.	Suggested	Select **Syntax Check** from the **Tools** menu, or click on the Check Syntax icon in the Toolbar. Assuming all the syntax in your script is correct, a status line at the bottom of the Author will display the message “Syntax check successful, with no errors.” If the syntax check fails, correct any errors and check the syntax of the script again until successful.	Tools menu or Check Syntax Toolbar icon
40.	Required	Save this script again, to ensure all changes you have made since the script was renamed at the beginning of this exercise are retained.	File menu *or* Script Author Toolbar *or* keyboard
41.	Required	From the Tools menu, select **Deploy Script**. Populate the appropriate fields to reflect your applications database location and deploy the script as before.	Deploy Script To dialog
42.	Required	Launch the Scripting Engine by using the prescribed manner for your environment, select **practice_script_no_Java_5**, from the Script Chooser, and launch the script. Note that when the script launches, the company name you added as a Constant value appears in a grey region above the active panel. Also note that when you reach the Goodbye panel, the name entered into the Hello panel appears as spoken text.	Script Engine

Concepts Demonstrated in Exercise B-5

Creating Commands

Regardless of the type of command, the method to associate a command with the global script, as a pre- or post-action of a panel, or as an action is the same. In this exercise, the two command types that can be executed without custom code were used. There are a variety of ways to use *blackboard* values and *constants*; these are only samples to provide a simple illustration of commands in Scripting.

For more information on using commands or other aspects of Oracle Scripting, refer to the *Oracle Scripting Concepts and Procedures* document.

C

Scripting Project Scoping and Discovery

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 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 to an existing script or call guide that the customer wants. It is important to freeze requirements early to ensure deadlines are met.

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, 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 customer and the consultants understand and map out all business rules covering every eventuality. Oracle Scripting is an excellent tool to assist agents in presenting information logically and consistently, but the tool alone is not a guarantee of success. A clear understanding of the business requirements is key. In order to satisfy both the customer and the consulting team, detailed flowcharting of the

entire script and all its possible branches should be performed. This may require specifically trained business analysts and should be a prerequisite of accepting a scripting engagement, or at the very least, must be completed (and agreed upon by both parties) prior to initiating development.

This point bears repeating: if you do not know all requirements of a script prior to accepting the engagement, there is no clear acceptance criteria to determine when the project is complete. It is to the benefit of customers implementing Scripting as much as to Oracle consultants or partners 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 customer 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 customer's interest to follow the progress of the development of the initial script to be delivered, so they 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 customer) is greatly simplified—whether performed by the customer staff alone, or with additional consulting support from Oracle consultants or partners.

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 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 Database / API Blocks
- 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
- Populating the global 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 databases
- Reusing functionality in existing scripts as Templates
- Integrating and modifying Best Practices Scripts

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 Scripting approach is best served by breaking up a question into two parts.

As mentioned in the Scripting Implementation Project Scoping section in the main body of the implementation guide above, the Text Layer can provide certain challenges, for example in the case where a call guide has undergone legal review and requires approval for any changes.

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 customer when broken into a phased approach. In this way, customers 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 customer can have the interaction center up and running using Oracle Scripting, taking the opportunity to train agents and staff while experiencing the advantages of the efficiencies and return on investment of automated scripting.

Tools to Aid in Scoping and Discovery

As discussed in the Planning an Oracle Scripting Implementation section above, appropriate planning is crucial to the success of a Scripting implementation. Tools to help the Oracle consultant and Oracle customers include the 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. (See AIM's Role in Scripting Implementation Planning above for more information on AIM.)

Part of a consultant's skill set is the ability to perform a thorough Discovery process to fully understand a customer's requirements, existing infrastructure and environment, and long-term needs. Included in this appendix are some tools developed in the Consulting organization as an aid to scoping a potential Scripting engagement. These include the Oracle Scripting Discovery Data Worksheet, which helps to gather information specific to Scripting that should be obtained for the potential Scripting engagement in general. The other tool provided here is the Oracle Scripting Discovery Checklist, which should be filled out for each distinct functionality expected to be created in a script (for example, each group on the canvas).

Using these as aids to the Discovery process can help gather information that will be crucial to appropriate planning, allocation of resources, and scoping of the effort in general.

Oracle Scripting Discovery Data Worksheet

The following worksheet covers an entire Scripting engagement for which you are attempting to gauge the scope in order to properly provide a quote for labor and to assess the skill and resource requirements.

1. For which of the following purposes does the customer / prospect intend to use Oracle Scripting?

   ☐ Pure scripting purposes ☐ Desktop integration tool ☐ Both ☐ Undetermined

   If the customer wishes to use Oracle Scripting for desktop integration, describe the requirements in general terms:

2. Characterize the Scripting required:

   ☐ Inbound Call Guide ☐ Outbound Call Guide ☐ Both

   ☐ Guided Selling ☐ Undetermined

3. Does the customer require call guides that support languages other than English?

   ☐ Yes ☐ No If yes, identify the language(s):

4. How many call guides or individual scripts does the customer want Oracle (or Oracle partners) to develop? If undetermined, please indicate.

5. Does the customer currently have existing scripts that they wish to convert to Oracle Scripting?

   ☐ Yes ☐ No If yes, how many?

If appropriate, attach the existing scripts to this document and indicate form of existing scripts:

Attached as: ☐ Hardcopy ☐ Electronic File ☐ Not Available

6. What call guide or script currently exists from which Oracle Scripting scripts will be developed?

☐ Narrative ☐ Flowcharting ☐ System Requirements Document (SRD) ☐ Existing scripts

7. Is Computer-Telephony Integration (CTI) intended for use in the facility or facilities?

☐ Yes ☐ No

Does the customer have a need to display different call guides using CTI?

☐ Yes ☐ No

8. If so, what will be the criteria?

☐ Dialed Number Information Service (DNIS) ☐ Automatic Number Identification (ANI)

☐ Unique ID (UUID) ☐ Account type ☐ IVR information ☐ Other

If Interactive Voice Response (IVR) unit information required, is IVR in place? ☐ Yes ☐ No

9. Does CTI currently exist? ☐ Yes ☐ No Describe:

10. Does the customer plan to create and maintain its own call guides or scripts? ☐ Yes ☐ No

11. **Gap Analysis.** Describe in detail any modifications that Oracle (or partner) must make to the generic version of Oracle Scripting in order to satisfy the needs of the customer:

Oracle Scripting Discovery Checklist Tool

First, qualify each call guide or script required using the following checklist. Then, 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 _____ ☐ Main script checklist ☐ Script functionality breakdown checklist

Number of Panels:		***Development Level Assessment***
☐ Very Small:	1 to 10 panels	☐ Very Simple
☐ Small:	11 to 25 panels	☐ Simple
☐ Medium:	25 to 40 panels	☐ Lower Intermediate
☐ Medium to Large:	40 to 100 panels	☐ Upper Intermediate
☐ Large:	100 to 200 panels	☐ Complex
☐ Very Large:	Over 200 panels	☐ 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:

Custom PL/SQL	Currently exists?	☐ Y	☐ N	Amount:	☐ 1 or 2	☐ Several	☐ Many (List)
Custom Java	Currently exists?	☐ Y	☐ N	Amount:	☐ 1 or 2	☐ Several	☐ Many (List)
Blackboard	Currently exists?	☐ Y	☐ N	Amount:	☐ 1 or 2	☐ Several	☐ Many (List)
Other (list)	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:**

D

Modifying appsweb.cfg File for Scripting Implementations

Description of appsweb.cfg File

The appsweb.cfg file is the Oracle Forms Web CGI configuration file for Oracle Applications 11*i*. This file defines parameter values used by the Forms Web CGI on the Web server. These parameter values are defined upon installation during the Rapid Install.

Among its parameters, the appsweb.cfg file determines several categories of Java Archive files. For the purposes of this implementation guide, the two categories of concern are those designated as "core" JAR files (used by all Forms-based products), and those that can be downloaded upon demand (for example, by selecting a particular Oracle Applications responsibility and launching a particular Oracle business application).

Understanding Core Files and On-Demand Files

When Various JAR Files Are Downloaded

When an agent logs into Oracle Applications, Oracle JInitiator runs the Oracle Forms Java applet, caching the core JAR files to the client.

On-demand files are only downloaded when required, as designated by the Java Class Loader. When an 11*i* Oracle Application needs to reference Java methods in a particular class, this represents a demand for the file, which is then downloaded from its location on the web server.

Relevance to Oracle Scripting

While the `appsweb.cfg` file has several uses, this document discusses only those that are relevant to implementations of Oracle Scripting. Accordingly, only the physical structures of this file relevant to Scripting are discussed herein.

Format of `appsweb.cfg` file

Files listed in an archive within the `appsweb.cfg` file are characterized as core or on-demand by their placement in this configuration file. Essentially, any file (in any archive) that *precedes* the `fndlist.jar (,/OA_JAVA/oracle/apps/fnd/jar/fndlist.jar,)` is a core file. Any file *following* the `fndlist.jar` (even in the same archive) is loaded on demand.

JAR files listed in the `appsweb.cfg` file are arranged into logical blocks called *archives*. Each archive is labeled, (for example, `archive1, archive2`, etc.). The archive structure is provided for the convenience of grouping JAR files into categories. For example, in the sample below, JAR files to support CRM products are primarily listed in `archive3`.

JAR files listed to support Apple Macintosh operating systems are included as separate archives (for example, `macarchive1, macarchive2`, etc.) Thus, you should duplicate the modifications you make to the archive blocks above to the macharchives, to ensure those classes are available in all cases.

Configuring the `appsweb.cfg` File for the Caching Architecture

At the current time, the configuration file must be modified for three reasons:

1. To appropriately reference custom Java code in the core classes. This step ensures that custom Java methods written in support of a specific script are downloaded upon Forms login (login to Oracle Applications) to the agent desktop.
2. To include the JDBC driver in the core classes to support Oracle Scripting's use of thin JDBC to communicate between the client and the database.
3. To include AOL/J classes as load-on-demand. This set of classes has been excluded from the `appsweb.cfg` file.

Other Applications for Modifying the `appsweb.cfg` File

Moving reference to the three foundation Scripting JAR files and other Scripting-dependent files from their current location (in archive 3 in the `appsweb.cfg` file sample below) to precede the `fndlist.jar` is a method to

reduce the amount of time required to start Scripting. The downside to this customization of the configuration file is that those classes will download for *any* user who logs in through a Forms application. This is discussed in more detail below.

Sample `appsweb.cfg` File

Following is the header and then the relevant consecutive portions of a sample `appsweb.cfg` file that can be referenced when performing implementation steps.

```
; Forms Web CGI Configuration File for Oracle Applications 11i
; $Header: appsweb.cfg 115.13 2000/08/11 16:58:59 pkm ship     $
; ----------------------------------------------------------
. . .
; JAR Files
; ---------
; Client java code is distributed via JAR files.
; The order of jar files listed is important, as on-demand-loading is the
; default. For more JAR file loading options see below.
;
; Core JAR files used by all Forms-based products
archive=/OA_JAVA/oracle/apps/fnd/jar/fndforms.jar,/OA_JAVA/oracle/apps/fnd/jar/
fndformsi18n.jar,/OA_JAVA/oracle/apps/fnd/jar/fndewt.jar,/OA_JAVA/oracle/apps/
fnd/jar/fndswing.jar,/OA_JAVA/oracle/apps/fnd/jar/fndbalishare.jar,/OA_JAVA/
oracle/apps/fnd/jar/fndaol.jar,/OA_JAVA/oracle/apps/fnd/jar/fndctx.jar
;
; The following JAR files are loaded on demand
; JAR files used for FND products
archive1=,/OA_JAVA/oracle/apps/fnd/jar/fndlist.jar,/OA_JAVA/oracle/apps/fnd/jar/
fndnetcharts.jar,/OA_JAVA/oracle/apps/fnd/jar/fndtdg.jar,/OA_JAVA/oracle/apps/
.  .  .
.  .  .
; JAR files for CRM products  (the list starts with a comma)
archive3=,/OA_JAVA/oracle/apps/asg/jar/asg.jar,/OA_JAVA/oracle/apps/asg/jar/
asgmtran.jar,/OA_JAVA/oracle/apps/ast/jar/asthgrid.jar,/OA_JAVA/oracle/apps/
ast/jar/astuijav.jar,/OA_JAVA/oracle/apps/cct/jar/cctotm.jar,/OA_JAVA/oracle/
apps/cct/jar/cctnet.jar,/OA_JAVA/oracle/apps/cct/jar/cctsoft.jar,/OA_JAVA/
oracle/apps/cct/jar/cctroute.jar,/OA_JAVA/oracle/apps/csf/jar/csfmap.jar,/OA_
JAVA/oracle/apps/csf/jar/csfchart.jar,/OA_JAVA/oracle/apps/csf/jar/csflf.jar,
/OA_JAVA/oracle/apps/csf/jar/csftds.jar,/OA_JAVA/oracle/apps/csr/jar/
csrclient.jar,/OA_JAVA/oracle/apps/ibu/jar/ibu.jar,/OA_JAVA/oracle/apps/iem/
jar/iemsrv.jar,/OA_JAVA/oracle/apps/iem/jar/iemadm.jar,/OA_JAVA/oracle/apps/
iem/jar/iemegen.jar,/OA_JAVA/oracle/apps/iem/jar/iemedit.jar,/OA_JAVA/oracle/
```

apps/iem/jar/iemclnt.jar,/OA_JAVA/oracle/apps/iem/jar/iemapplt.jar,/OA_JAVA/oracle/apps/ies/jar/iescommn.jar,/OA_JAVA/oracle/apps/ies/jar/iesclien.jar,/OA_JAVA/oracle/apps/ies/jar/iesservr.jar,/OA_JAVA/oracle/apps/ieu/jar/ieunet.jar,/OA_JAVA/oracle/apps/ieu/jar/ieustuba.jar,/OA_JAVA/oracle/apps/ieu/jar/ieuui.jar,/OA_JAVA/oracle/apps/ieu/jar/ieuclient.jar,/OA_JAVA/oracle/apps/ieu/jar/ieucommon.jar,/OA_JAVA/oracle/apps/ieu/jar/ieuutil.jar,/OA_JAVA/oracle/apps/ieu/jar/ieutrans.jar,/OA_JAVA/oracle/apps/iex/jar/iexdbjav.jar,/OA_JAVA/oracle/apps/iex/jar/iexbeans.jar,/OA_JAVA/oracle/apps/jtf/jar/jtfui.jar,/OA_JAVA/oracle/apps/jtf/jar/jtfgrid.jar,/OA_JAVA/oracle/apps/jtf/jar/jtfgantt.jar,/OA_JAVA/oracle/apps/xnp/jar/xnpadptr.jar,/OA_JAVA/oracle/apps/csc/jar/csc.jar

Generic Instructions for Modifying `appsweb.cfg` File

1. Stop your Web listeners and Forms server.
2. Identify the appsweb.cfg version of any backup copy you might have in $FND_TOP/resource or $OA_HTML/bin. For example:
   ident $OA_HTML/bin/appsweb.cfg*
3. Make a backup copy of current $OA_HTML/bin/appsweb.cfg and $FND_TOP/resource/appsweb.cfg. For example:
   cp $OA_HTML/bin/appsweb.cfg OA_HTML/bin/appsweb.cfg.pre_<bug#>.
4. Edit $OA_HTML/bin/appsweb.cfg, appending the name and path of the new JAR file you wish to add to (or move within) the appropriate location. Start each new file with a comma, following this format:

   ,/OA_JAVA/oracle/apps/<prod>/jar/<JAR-file>.jar
5. Note that $FND_TOP/resource/appsweb.cfg is supposed to be the master copy of appsweb.cfg, but $OA_HTML/bin/appsweb.cfg is used at runtime. *Only if you are very familiar with the appsweb.cfg file,* compare and consolidate the config files $OA_HTML/bin/appsweb.cfg and $FND_TOP/resource/appsweb.cfg (and backups) to resolve any discrepancies.
6. If unable to resolve discrepancies, contact Oracle Support Services. To obtain a working version while resolving the discrepancies, copy $OA_HTML/bin/appsweb.cfg to $FND_TOP/resource/appsweb.cfg.
7. Restart the Forms server.
8. Restart the web listener serving Forms Applets (WebDB 2.2 in Oracle Applications 11.5.1). Make sure you have sourced your Applications environment, so that environment variable FORMS60_WEB_CONFIG_FILE is defined.
9. Start Oracle Applications 11*i*, choose a Forms-based responsibility, and verify that modifications have had the expected result.

Modifications for Caching Architecture

For Caching Architecture operations (regardless of whether you are implementing from scratch or converting a 3-tier JServer Technology stack implementation to the Caching Architecture), three changes are required to be made to the `appsweb.cfg` file:

1. Reference the full path of any custom JAR files, designating them as core files by listing them prior to the `fndlist.jar` file. For more information, see Deploying Custom Code to the APPL_TOP
2. Reference the driver for using thin JDBC protocol (jdbc111.zip) as a core file (again, list this file prior to the fndlist.jar file).
3. Reference the fndaolj.jar file as an on-demand file.

The first two changes should be made to the core archive of the configuration file so that the custom Java files are downloaded to the client upon launching Oracle Applications, making the driver available.

The remaining change (`fndaolj.jar`) should ensure that this file is listed after the `fndlist.jar` file, designating it as an on-demand file. In the sample below, this file is appended to the archive that refers to CRM applications.

Sample Modified `appsweb.cfg` File

Following are the relevant portions of a sample `appsweb.cfg` file that can be referenced when performing implementation steps.

After modifying the `appsweb.cfg` file to include the changes discussed above, the customized file should appear as indicated below.

Note: The code below is just a sample, and some elements for your configuration may differ. What is illustrated here is that the files in red were added to the previously shown sample configuration file. The fndlist.jar file that separates core from on-demand files is indicated in **`bold print`**.

```
; Forms Web CGI Configuration File for Oracle Applications 11i
; $Header: appsweb.cfg 115.13 2000/08/11 16:58:59 pkm ship     $
; ----------------------------------------------------------
. . .
; JAR Files
; ---------
; Client java code is distributed via JAR files.
; The order of jar files listed is important, as on-demand-loading is the
; default. For more JAR file loading options see below.
;
; Core JAR files used by all Forms-based products
archive=/OA_JAVA/oracle/apps/fnd/jar/fndforms.jar,/OA_JAVA/oracle/apps/fnd/jar/
fndformsi18n.jar,/OA_JAVA/oracle/apps/fnd/jar/fndewt.jar,/OA_JAVA/oracle/apps/
fnd/jar/fndswing.jar,/OA_JAVA/oracle/apps/fnd/jar/fndbalishare.jar,/OA_JAVA/
oracle/apps/fnd/jar/fndaol.jar,/OA_JAVA/oracle/apps/fnd/jar/fndctx.jar,/OA_
JAVA/ies_custom/custom.jar,/OA_JAVA/jdbc111.zip
;
;;
; The following JAR files are loaded on demand
; JAR files used for FND products
archive1=,/OA_JAVA/oracle/apps/fnd/jar/fndlist.jar,/OA_JAVA/oracle/apps/fnd/jar/
fndnetcharts.jar,/OA_JAVA/oracle/apps/fnd/jar/fndtdg.jar,/OA_JAVA/oracle/apps/
.  .  .
.  .  .
; JAR files for CRM products  (the list starts with a comma)
archive3=,/OA_JAVA/oracle/apps/asg/jar/asg.jar,/OA_JAVA/oracle/apps/asg/jar/
asgmtran.jar,/OA_JAVA/oracle/apps/ast/jar/asthgrid.jar,/OA_JAVA/oracle/apps/
ast/jar/astuijav.jar,/OA_JAVA/oracle/apps/cct/jar/cctotm.jar,/OA_JAVA/oracle/
```

apps/cct/jar/cctnet.jar,/OA_JAVA/oracle/apps/cct/jar/cctsoft.jar,/OA_JAVA/oracle/apps/cct/jar/cctroute.jar,/OA_JAVA/oracle/apps/csf/jar/csfmap.jar,/OA_JAVA/oracle/apps/csf/jar/csfchart.jar,/OA_JAVA/oracle/apps/csf/jar/csflf.jar,/OA_JAVA/oracle/apps/csf/jar/csftds.jar,/OA_JAVA/oracle/apps/csr/jar/csrclient.jar,/OA_JAVA/oracle/apps/ibu/jar/ibu.jar,/OA_JAVA/oracle/apps/iem/jar/iemsrv.jar,/OA_JAVA/oracle/apps/iem/jar/iemadm.jar,/OA_JAVA/oracle/apps/iem/jar/iemegen.jar,/OA_JAVA/oracle/apps/iem/jar/iemedit.jar,/OA_JAVA/oracle/apps/iem/jar/iemclnt.jar,/OA_JAVA/oracle/apps/iem/jar/iemapplt.jar,/OA_JAVA/oracle/apps/ies/jar/iescommn.jar,/OA_JAVA/oracle/apps/ies/jar/iesclien.jar,/OA_JAVA/oracle/apps/ies/jar/iesservr.jar,/OA_JAVA/oracle/apps/ieu/jar/ieunet.jar,/OA_JAVA/oracle/apps/ieu/jar/ieustuba.jar,/OA_JAVA/oracle/apps/ieu/jar/ieuui.jar,/OA_JAVA/oracle/apps/ieu/jar/ieuclient.jar,/OA_JAVA/oracle/apps/ieu/jar/ieucommon.jar,/OA_JAVA/oracle/apps/ieu/jar/ieuutil.jar,/OA_JAVA/oracle/apps/ieu/jar/ieutrans.jar,/OA_JAVA/oracle/apps/iex/jar/iexdbjav.jar,/OA_JAVA/oracle/apps/iex/jar/iexbeans.jar,/OA_JAVA/oracle/apps/jtf/jar/jtfui.jar,/OA_JAVA/oracle/apps/jtf/jar/jtfgrid.jar,/OA_JAVA/oracle/apps/jtf/jar/jtfgantt.jar,/OA_JAVA/oracle/apps/xnp/jar/xnpadptr.jar,/OA_JAVA/oracle/apps/csc/jar/csc.jar,/OA_JAVA/oracle/apps/fnd/jar/fndaolj.jar

Pre-Loading Scripting .jar Files to Reduce Scripting Startup Time

Some .jar files (those designated as core as indicated by their location in the appsweb.cfg file) are always loaded at the time of the Forms login. Other .jar files are only loaded when needed (on-demand). All .jar files initially reside on the Web server and are referenced in the appsweb.cfg file.

The first time a user logs into an Oracle Forms session, Oracle JInitiator downloads and caches the core .jar files (specified in appsweb.cfg) locally on the client machine. For each subsequent Oracle Forms session, as long as the specified core .jar files on the Web server have not changed, they are loaded into memory (the Class Loader within the JVM) from the cached files on the local disk.

From an Oracle Scripting perspective, most files required specifically for Scripting are designated in appsweb.cfg as on-demand files. Each time a script is requested, Oracle JInitiator compares the time stamps of those .jar files in cache with those on the Web server. If files have been updated on the Web server, the new files will be cached to the client and loaded into memory; otherwise, the cached files will be loaded into memory.

While it is true that Scripting launches more quickly in second and subsequent script sessions (providing updated files have not been uploaded), the time-stamp comparison process and the process of loading the .jar files into memory (even the cached files) consumes a significant amount of time.

This amount of time cannot be reduced; however, the impact to customers *can* be. The time-stamp comparison and loading of required .jar files into memory occurs the first time (per Forms session) that an agent invokes a script. Thus, for each agent's Oracle Applications session, customers may be forced to wait the amount of time needed for these processes to take place.

By modifying the appsweb.cfg file, the on-demand files required for Scripting can be changed to core files. JInitiator then performs the time-stamp comparison and loading of required .jar files into the Class Loader of the JVM *at the beginning of each Forms session* (specifically, when an agent logs into Oracle Applications). Thus, even though the agent must still wait for these processes, customers are not impacted.

Determining Which Files Must Be Moved in appsweb.cfg

Note that this modification is only recommended after an implementation has been completed, tested, and you are certain all aspects are functioning as you wish. Then, launch the specific script with the Oracle JInitiator Java console active. View and make note of the .jar files being loaded in the Java console to ensure that you know which files are needed for Oracle Scripting and which custom .jar files are required to support this particular script.

To Where Should Files Must Be Moved in appsweb.cfg

The .jar files you noted when Scripting launched should be moved from their current location in the appsweb.cfg file (following the fndlist.jar file) to a location preceding the findlist.jar file. The order in which the files are listed prior to the fndlist.jar file is not relevant.

Ramifications

Note that any user logging into a Oracle Applications 11*i* Forms application at the enterprise (including those who do not need to use Scripting) will automatically load the Scripting .jar files as part of the Forms login. The modifications discussed here do not change the overall time to load scripts, but rather moves the time to load the .jar files to the Forms login rather than the Scripting startup.

Disclosure

This modification is not recommended by Oracle Corporation, but is described here to fully inform users of Oracle Scripting. Trade-offs should be understood and analyzed before such modifications are implemented.