Oracle® Self–Service Web Applications Implementation Manual

Release 11

March 1998

ORACLE®

Enabling the Information Age™

Oracle® Self–Service Web Applications Implementation Manual
Release 11

The part number for this volume is A58294–01.

Major Contributors: Kurt Thompson, Michelle Jacobsen, Neal Barlow, Kevin Wiggen, Theresa Chai, Liza Broadbent, Rammi Haddad, Desmond Chu

Contributors: John Houghton

Contents

Preface

Welcome to the *Oracle Self–Service Web Applications Implementation Manual*.

This manual includes information you need to work with Oracle Self–Service Web Applications effectively. It contains detailed information about the following:

- Overview and architecture
- Setup
- Oracle Web Applications Dictionary overview and procedures
- Predefined inquiry flows
- Application Programmable Interfaces (API)

This preface explains how this user's guide is organized and introduces other sources of information that can help you.

About this User’s Guide

This guide contains overviews as well as task and reference information about Oracle Self–Service Web Applications. This guide includes the following chapters:

- Chapter 1 presents an overview of Oracle Self–Service Web Applications, including its architecture, data security, and how it relates to Oracle Applications.
- Chapter 2 describes how to set up Oracle Self–Service Web Applications.
- Chapter 3 describes the Oracle Web Applications Dictionary and how to use it.
- Chapter 4 provides an overview of the predefined inquiry flows that ship with Oracle Self–Service Web Applications.
- Chapter 5 describes the Application Programmable Interfaces.

Audience for This Guide

This guide assumes you have a working knowledge of your business area's processes and tools. It also assumes you are familiar with Self–Service Web Applications. If you have never used Self–Service Web Applications, we suggest you attend one or more of the Self–Service Web Applications training classes available through Oracle Education Services. For more information about Self–Service Web Applications and Oracle training, see: Other Information Sources.

Do Not Use Database Tools to Modify Oracle Applications Data

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

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

Consequently, we STRONGLY RECOMMEND that you never use SQL*Plus or any other tool to modify Oracle Applications data unless otherwise instructed.

Other Information Sources

Here are some other ways you can increase your knowledge and understanding of Self–Service Web Applications.

Online Documentation

All Oracle Applications documentation is available online on CD–ROM, except for technical reference manuals. There are two online formats, HyperText Markup Language (HTML) and Adobe Acrobat (PDF).

All user's guides are available in HTML, Acrobat, and paper. Technical reference manuals are available in paper only. Other documentation is available in Acrobat and paper.

The *content* of the documentation does not differ from format to format. There may be slight differences due to publication standards, but such differences do not affect content. For example, page numbers and screen shots are not included in HTML.

The HTML documentation is available from all Oracle Applications windows. Each window is programmed to start your web browser and open a specific, context–sensitive section. Once any section of the HTML documentation is open, you can navigate freely throughout all Oracle Applications documentation. The HTML documentation also ships with Oracle Information Navigator (if your national language supports this tool), which enables you to search for words and phrases throughout the documentation set.

Related User's Guides

Self–Service Web Applications shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user's guides when you set up and use Self–Service Web Applications.

If you do not have the hardcopy versions of these manuals, you can read them online using the Applications Library icon or Help menu command.

Oracle Applications User's Guide

This guide explains how to enter data, query, run reports, and navigate through Oracle Applications products. This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

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

Oracle Applications Demonstration User's Guide

This guide documents the functional storyline and product flows for Global Computers, a fictional manufacturer of personal computers products and services. As well as including product overviews, the book contains detailed discussions and examples across each of the major product flows. Tables, illustrations, and charts summarize key flows and data elements.

Reference Manuals

Oracle Automotive Implementation Manual

This manual describes the setup and implementation of the Oracle Applications used for the Oracle Automotive solution.

Oracle Manufacturing, Distribution, Sales and Service Open Interfaces Manual

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

Oracle Applications Message Reference Manual

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

Oracle Project Manufacturing Implementation Manual

This manual describes the setup steps and implementation for Oracle Project Manufacturing.

Installation and System Administration

Oracle Alert User's Guide

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

Multiple Reporting Currencies in Oracle Applications

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

Multiple Organizations in Oracle Applications

If you use the Oracle Applications Multiple Organization Support feature to use multiple sets of books for one Self–Service Web Applications installation, this guide describes all you need to know about setting up and using Self–Service Web Applications with this feature.

Oracle Applications Implementation Wizard User's Guide

If you are implementing more than one Oracle product, you can use the Oracle Applications Implementation Wizard to coordinate your setup activities. This guide describes how to use the wizard.

Oracle Applications Developer's Guide

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards.* It also provides information to help you build your custom Developer/2000 forms so that they integrate with Oracle Applications.

Oracle Applications Flexfields Guide

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

Oracle Applications Installation Manual for Windows Clients

This guide provides information you need to successfully install Oracle Financials, Oracle Public Sector Financials, Oracle Manufacturing, or Oracle Human Resources in your specific hardware and operating system software environment.

Oracle Applications Product Update Notes

If you are upgrading your Oracle Applications, refer to the product update notes appropriate to your update and product(s) to see summaries of new features as well as changes to database objects, profile options and seed data added for each new release.

Oracle Applications Upgrade Preparation Manual

This guide explains how to prepare your Oracle Applications products for an upgrade. It also contains information on completing the upgrade procedure for each product. Refer to this manual and the *Oracle Applications Installation Manual* when you plan to upgrade your products.

Oracle Applications System Administrator's Guide

This manual provides planning and reference information for the Self–Service Web Applications System Administrator.

Other Sources

Training

We offer a complete set of formal training courses to help you and your staff master Self–Service Web Applications. We organize these courses into functional learning paths, so you take only those courses appropriate to your job or area of responsibility.

You have a choice of educational environments. You can attend courses offered by Oracle Education Services at any one of our Education Centers, or you can arrange for our trainers to teach at your facility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization structure, terminology, and data as examples in a customized training session delivered at your own facility.

Support

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

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 45 software modules for financial management, supply chain management, manufacturing, project systems, human resources, and sales and service management.

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

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 140 countries around the world.

Thank You

Thank you for using Self–Service Web Applications and this implementation manual.

We value your comments and feedback. At the end of this manual is a Reader's Comment Form you can use to explain what you like or dislike about Self–Service Web Applications or this manual. Mail your comments to the following address or call us at (650) 506–7000.

Oracle Applications Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Or, send electronic mail to **appsdoc@us.oracle.com**.

CHAPTER

1

Overview of Self–Service Web Applications

This chapter presents an overview of Self–Service Web Applications, including a discussion of the following topics:

- Oracle Self–Service Web Applications Architecture: page 1 – 3
- Data Security: page 1 – 15

Overview

The Oracle Self–Service Web Applications, consisting of Web Customers, Web Employees, and Web Suppliers, are extensions to Oracle Applications. These self–service applications extend, but do not duplicate the functionality of Oracle Applications. Oracle Web Customers, Web Employees, and Web Suppliers adds a browser–based, walk up and use functionality that supplements Oracle Applications.

The self–service web applications can be either inquiry or transactional. Inquiry modules read but do not update the Oracle Applications database; transactional modules do update the database.

See Also

Oracle Self–Service Web Applications Architecture: page 1 – 3

Data Security: page 1 – 15

Predefined Inquiry Flows: page 4 – 2

Oracle Self–Service Web Applications Architecture

The architecture consists of the following components:

- a web browser
- the Web Listener
- HTML documents
- the Web Request Broker
- the Common Gateway Interface (CGI).

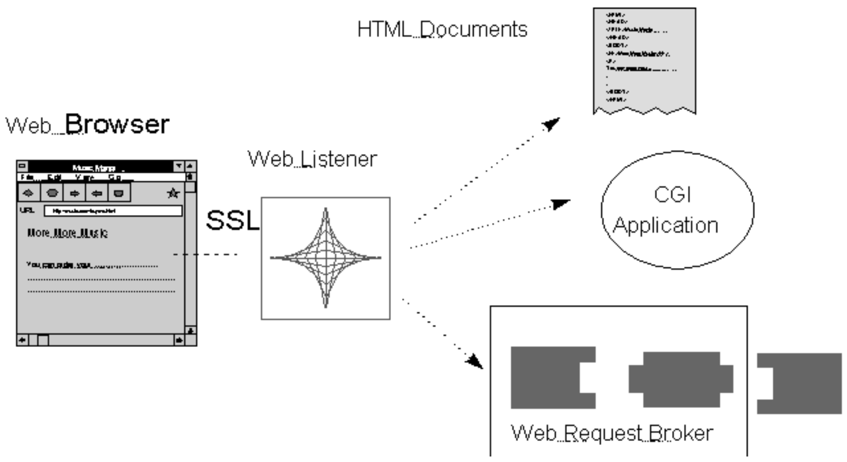


Figure 1 – 1
Product Architecture

See the detailed sections below:

- Web Request Broker Architecture: page 1 – 7
- PL / SQL Agent: page 1 – 8
- Oracle Workflow: page 1 – 10
- Web Applications Dictionary: page 1 – 10
- Web Inquiries and Web Transactions: page 1 – 11

The following definitions will help you to understand the big picture of Oracle Self–Service Web Applications.

Definitions

Common Gateway Interface (CGI)

The industry standard technique for running applications on a web server. Oracle Web Application Server supports this standard and offers additional functionality with the Web Request Broker.

Flow

A series of web pages, each of which can display data. The pages that make up a flow are bound together by complex definitions. Specifically, flows are comprised of pages, page regions, and region items.

HTML (HyperText Markup Language)

A format for encoding hypertext documents that may contain text, graphics, and references to programs, and references to other hypertext documents. HTML is a subset of Standard Generalized Markup Language (SGML).

HTTP (HyperText Transfer Protocol)

A protocol used to request documents from the web server.

Javascript

Javascript is a scripting language that adds significant power to HTML files without the need for server–based CGI programs.

Web Applications Dictionary

An active data dictionary that employs the Oracle Forms–based interface. The data dictionary stores specific information about Self–Service Web Applications data, including prompts, language, navigation, and security.

Web Request Broker (WRB) Cartridges

WRB applications, implemented as a shared library, uses the WRB API to handle HTTP requests from web clients. Cartridges can be developed in any language and then integrated with the WRB through the WRB API.

Web Browser

The client user interface component. The browser you use must support tables and frames and be Javascript enabled. The embedded Javascript code in Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers provides a mechanism for client side caching of user–entered data during a transaction, and simple client side validation of user–entered data. Execution of simple Javascript code logic at the client side results in reduced network traffic between the web browser client and the web server.

Oracle Web Application Server

Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers, Release 11 requires Oracle Web Application Server 3.0 or above. Web Application Server 3.0 is Oracle's standard HTTP server that offers an open, extensible application development platform for the Web. This enables secure, high speed transactions with the Oracle8 RDBMS.

Oracle Web Application Server is designed to be a scalable, high–performance network daemon. It is implemented as a multi–threaded, single–process asynchronous engine, allowing multiple concurrent requests to be processed at the same time using standard HTTP or HTTP over SSL.

For further information, refer to your *Oracle Web Application Server 3.0 Installation Guide* and other online documentation for Web Application Server 3.0.

Web Listener

A component of Oracle Web Application Server responsible for delivering incoming requests to the Web Request Broker, where they are processed or dispatched to services built with the Web Application Server Software Development Kit (SDK). The HTTP server is the network layer component of Oracle Web Application Server. It listens for incoming HTTP requests, delivers static files, and runs simple CGI programs. The HTTP server transfers everything else to the Web Request Broker (WRB), where a request is handled by a server extension.

The first generation of web sites on the Internet only offered static HTML files to their visitors. To enable dynamic content, web designers usually use CGI to extend the capabilities of their web server. Although easy to use, CGI has limitations, primarily the high overhead of spawning a new process for each incoming request, which causes degradation in scalability and throughput.

Currently, several web servers have defined an API for implementing extensions to the basic HTTP engine. The typical HTTP API enables a custom extension to be linked into the HTTP server process and allows it to intercept and fulfill a client request. Examples of such APIs are Netscape NSAPI, Spyglass ADI, and Microsoft IS–API. These APIs are becoming increasingly popular because they address the shortcomings of the original CGI protocol.

To compensate, Oracle Web Application Server 3.0 is layered into two fundamental components: an HTTP engine (Oracle Web Listener) and a high–speed dispatch mechanism (the Web Request Broker) that routes requests to server extensions running in separate processes.

Web Request Broker (WRB)

An asynchronous request handler with an API (Application Program Interface) that enables it to interface dynamically and seamlessly to various back–end technologies called WRB cartridges. It provides an architecture that allows server–side web applications to run under any HTTP server to which the WRB has been ported.

WRB applications, called cartridges, take advantage of the WRB's multi–process architecture to get higher performance than ordinary CGI scripts. The WRB architecture also makes WRB cartridges extremely scaleable—they can handle small request loads economically and large request loads efficiently.

The cartridge used by Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers is the PL/SQL Agent. This cartridge executes PL/SQL commands stored in the database.

Other cartridges that are **not** used by Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers, but are available through Oracle Web Application Server 3.0 are mentioned here for informational purposes:

Java Cartridge	Enables you to execute Java on the server to generate dynamic web pages.
LiveHTML	Enables the inclusion of output from programs supported by your operating system in your web pages.
Custom Cartridges	Since the WRB uses an open API, you can write your own cartridges to use it. Currently, only the C language is used to write WRB cartridges. In the future, however, the WRB API will also be available in other languages.
Third–Party Cartridges	Various independent vendors write cartridges for the WRB's open API.

Web Request Broker Architecture

The WRB architecture consists of these components:

- The Dispatcher
- The Web Request Broker
- WRB cartridges
- The WRB application engine
- WRB execution instances (WRBXs)

A WRB cartridge is implemented as a shared library that uses the WRB API to handle HTTP requests from web clients. The Dispatcher is a program that provides a CORBA (Common Object Request Broker Agent) interface between web listeners, the Web Request Broker, and WRB cartridges. The Dispatcher manages WRB cartridges and the requests directed to them. When the Web Listener receives an HTTP request directed to a WRB cartridge, it forwards the request to the Dispatcher. The Web Listener uses its configuration data to map URLs to WRB cartridges.

When the Dispatcher receives a request, it determines the WRB cartridge to which the request is directed, and directs the WRB to allocate an execution instance (WRBX) of that cartridge. The Dispatcher then dispatches the request to that WRBX. An execution instance of a WRB cartridge is a process running a program composed of two parts: a copy of the WRB application engine, and the WRB cartridge shared library.

The WRB application engine is the executable program that implements the WRB API and runs within each WRBX. It provides the interface between WRB cartridges and the Dispatcher, directs WRB cartridge flow of control, and provides services for WRB cartridges to use.

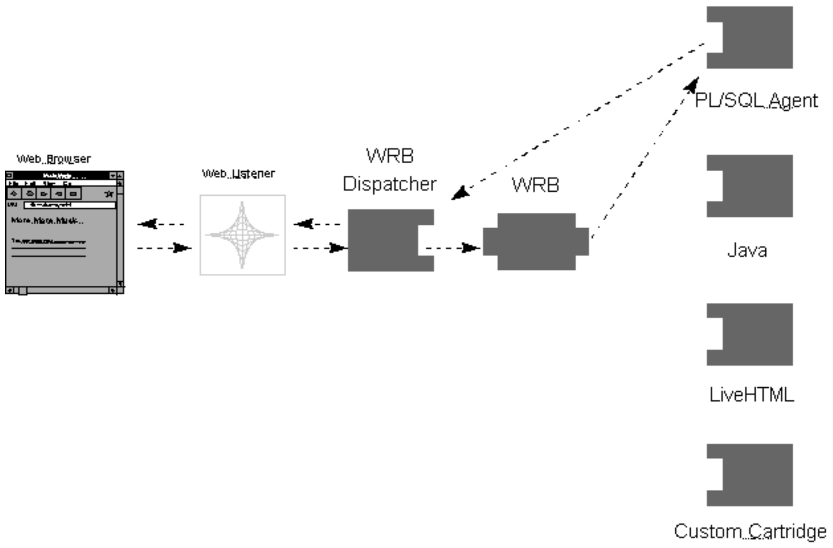


Figure 1 – 2
WRB cartridge instance creation in response to a request

The Dispatcher may then forward future requests directly to the cartridge instance and receive responses from the cartridge directly. The WRB configuration data specifies the maximum number of WRBXs that may run at once for each cartridge, and the minimum number of WRBXs that must always be running. You can tune the WRB performance by adjusting these values using the Web Request Broker administration pages.

PL/SQL Agent

The PL/SQL agent is responsible for connections to the database and executing stored procedures. When the Web Listener begins, it can start the WRBX (WRB Executable Engine) and interface with the PL/SQL agent based on how the Web Listener is configured. (The term PL/SQL agent refers to the WRBX interfacing with the PL/SQL agent cartridge.)

When the Web Listener receives a request for WRB, it forwards the request to the WRB dispatcher. The dispatcher works like a traffic director, redirecting requests to the appropriate cartridges. If the dispatcher determines that the URL is a request for the PL/SQL agent, the dispatcher forwards the URL to the PL/SQL agent.

Not only does the Dispatcher redirect requests, but it controls the number of WRBX processes running at any one time. If a request comes in for the PL/SQL agent and there is no WRBX running, the Dispatcher starts a WRBX. If many requests come in for the PL/SQL agent, the Dispatcher can distribute the load by starting multiple WRBX processes. If there are no requests coming in for a WRBX, the Dispatcher can shut down WRBX processes.

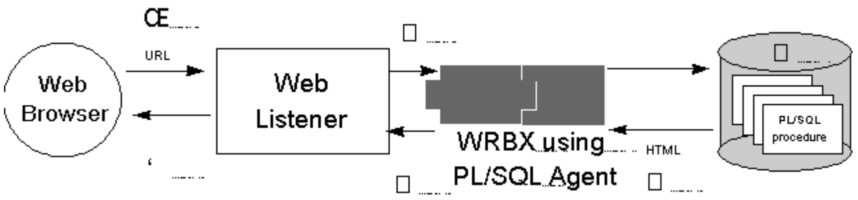


Figure 1 – 3
The web browser serving a dynamic page based on a PL/SQL request

1. User submits a URL from the browser that calls for the PL/SQL agent
2. Web Listener sends the request to WRB and on to the PL/SQL agent
3. PL/SQL agent logs in and invokes a PL/SQL procedure
4. PL/SQL procedure executes, generating an HTML document
5. PL/SQL agent passes the HTML document produced to the Web Listener and logs off. Session ends, but connection remains ready for next request.
6. Web Listener send the HTML document to the requesting browser

Any request starting with the directory path of /owa will be recognized and handed off to the PL/SQL agent. The syntax of a string calling a PL/SQL cartridge is

```
http://<machine>:<port>/<dcd_name>/<owa>/<stored procedure>
```

In the following example

```
http://19sun.us.oracle.com:9333/ic/owa/Oracle/OracleApps.VL
```

`19sun.us.oracle.com:9333` is the pointer to the web server which is listening on port 9333, `owa` is the directory path which has an entry in the WRB configuration file and indicates the request should route to the PL/SQL agent, and `OracleApps.VL` is the stored procedure, which is executed.

The DAD name specifies which database to connect to, what user to connect as, and how to connect to the database. Different DADs allow access to different databases or schema using the same Web Listener.

Each DAD specifies a username, password, database to connect to, and specifies how to connect to the database.

When it is called, the PL/SQL agent executes a stored procedure. This procedure is responsible for retrieving data and returning results in HTML format. By writing to an extension to the database standard output, an HTML document is generated via a PL/SQL stored procedure containing the appropriate data retrieved from the database. Within this generated page, there may be embedded Javascript code. Once the page is generated, it is returned to the client for display on a standard browser.

Oracle Workflow

Workflows can be defined for business flows so users can be sent automatically all the information they need to make a decision and have other business processes run automatically based upon their responses. See: *Oracle Workflow User's Guide, Release 11*.

Workflows are defined using the Workflow Builder, a Windows GUI interface that enables users to design the business process, the activities, items, messages and lookup lists, and roles (the approval chain). This workflow is then integrated into the business transaction process. For Web Employees, it is integrated with the requisition approval process.

Notifications generated in the workflow chain can be viewed with the Oracle Self–Service Web Applications or a Workflow–supported email system.

Oracle Web Employees includes a predefined workflow process to generate offer letters.

All workflow processes are customizable. See: *Oracle Workflow User's Guide, Release 11*.

Web Applications Dictionary

This is an Oracle Forms–based data dictionary used to define flow content and formatting for web inquiry pages. When users query for data, information is displayed on a web page, complete with hypertext links that enable the user to drill down to more detailed information. The pages that are linked in this way constitute a flow, alternatively referred to as an inquiry. Using the Web Applications Dictionary, you specify the content of, and links between the pages that make up a flow. Specifically, you can specify:

- HTML page format (headers, text, tables)

- Object content by associating with Applications Business Views or PL / SQL
- Business Flows among Objects (hypertext links)
- Page Content (fields, selection criteria)

Web Applications Dictionary also serves as a real time execution engine to retrieve information from the database. Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers reference the data dictionary at run time to retrieve data from the database and generate dynamic HTML pages.

The Web Applications Dictionary provides a means of defining business flows which can then be web–enabled using the existing functionality provided by Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers. All inquiry flows shipped with Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers were built using Web Applications Dictionary. These can be customized as needed.

The Web Applications Dictionary is part of Oracle Applications, Release 11, and is part of the named "AK Common Modules". Once installed, it is accessed in the same manner as all of the core Oracle Applications.

See: Web Applications Dictionary: page 3 – 2.

Web Inquiries and Web Transactions

Web Inquiries correspond to the query, or "read only", mode access to information stored in the Applications Server. Users are provided with a structured way of performing queries. The retrieved data is structured so that users can easily navigate through pages of closely linked information.

Web Transactions enable users to perform two simple transactions: place an order and enter a requisition. These transactions insert data into open interface tables. Data is then validated and then loaded into the core Oracle Applications production tables.

Web Inquiries

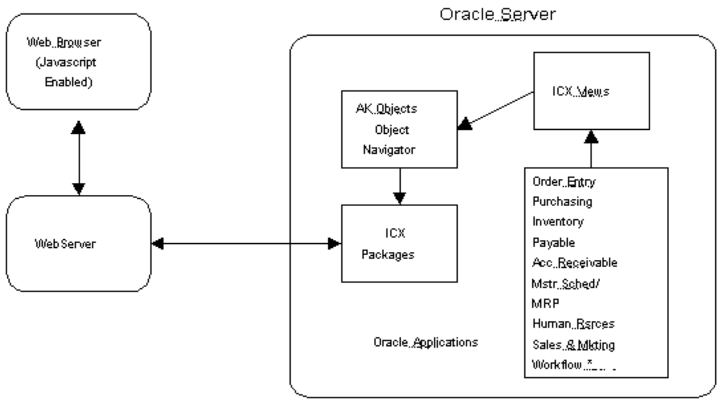


Figure 1 – 4
Web Inquiries

A web inquiry, or “flow”, is a series of hyperlinked web pages. Standard flows are predefined to allow users to easily navigate through web pages to access relevant information. These navigation flows are designed based on common business inquiry processes, and are built using Web Applications Dictionary. For example, a user can log in and request the View Purchase Orders inquiry. Once the data displays, the user can hyperlink to invoices and receipts related to the retrieved purchase orders.

Oracle Self–Service Web Applications (product code “ICX”) packages contain PL/SQL functions and procedures that access Web Applications Dictionary (product code “AK”) objects to retrieve information for the display elements and actual applications data. The Web Applications Dictionary has a run time execution engine that generates dynamic PL/SQL based on the ICX views. These views are based on those provided by the standard Oracle Applications.

Web Transactions

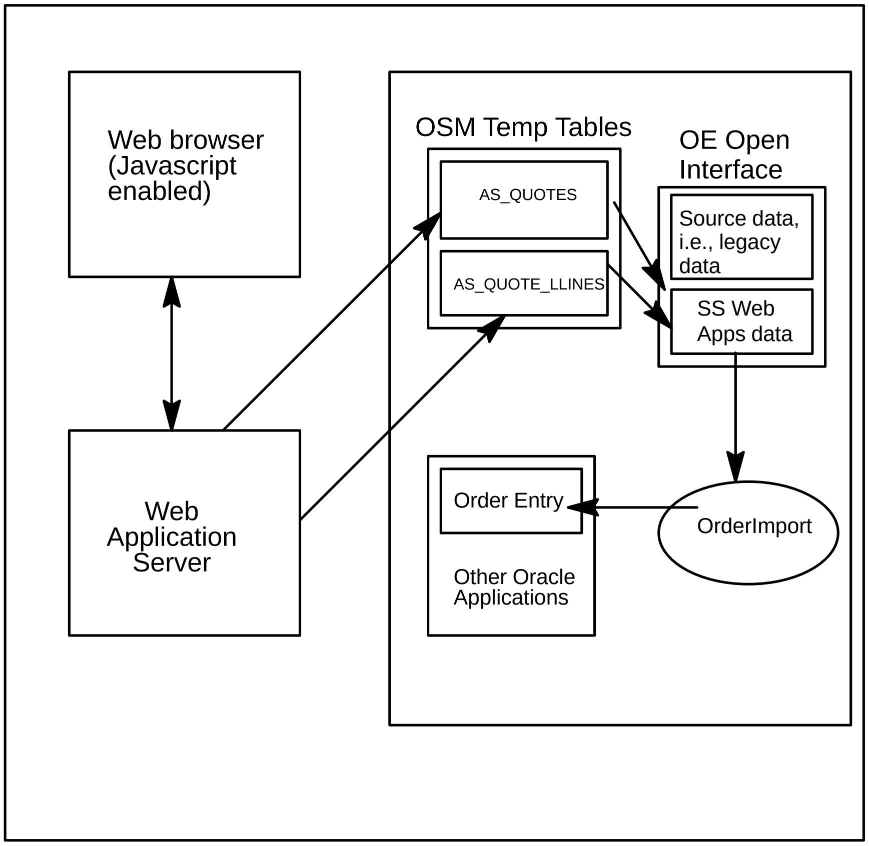


Figure 1 – 5
The process of performing a web transaction

The diagram above depicts the process of performing a web transaction (e.g., order entry) and the key components involved. A user who wants to enter a sales order uses a web browser to access the Web Store component of Oracle Customers. When the user enters lines on the order, the line data is stored at the client using Javascript technology. When the user submits the order, data is passed to the PL/SQL agent, which executes PL/SQL procedures to store the sales order data in the Oracle Sales and Marketing (OSM) shopping cart temporary tables. Finally, the data is loaded into the standard Order Entry Open Interface tables.

The batch interface OrderImport program in Oracle Order Entry then reads the open interface tables, performs necessary validations, and finally loads the data into Order Entry production tables. A source parameter can be set for the OrderImport program so that only imports those records entered via the web.

Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers only provide the logic to insert data into the open interface tables, leaving all validation logic to existing open interface programs.

The coding logic provided by Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers is mainly for building the user interface elements of the web. There is little transaction code; only limited Javascript logic for data caching at the client. Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers leverage the applications business logic provided by standard Oracle Applications by using its open interfaces.

Data Security

Data security for Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers is controlled by:

- Secure Socket Layers (SSL) to secure communication between client and server
- HTTP cookies
- encryption of password, parameter function, and session identifier
- session expiration
- securing and excluding attribute control

Session Management

Session management features include:

- each session is assigned a unique identifier, which is stored in a table
- session identifier returned to client encrypted via cookie
- encryption includes IP address to validate physical machine as well as the user
- session expiration based on number of hours or number of hits

Attribute Control

By using securing and excluding attributes, you can control user's access to data based on their ID and their responsibility. Attributes are first defined using the Web Applications Dictionary. They become securing or excluding attributes when you define responsibilities and users using the system administration functions of Oracle Application Object Library. See: Defining Attributes: page 3 – 25.

Securing Attributes for Row–Level Security

Securing attributes allow rows (records) of data to be visible to specified users or responsibilities based on the specific data (attribute value) contained in the row.

For example, to allow a hypothetical user, Sue, in the ADMIN responsibility to see rows containing a CUSTOMER_ID value of 1000, assign the securing attribute of CUSTOMER_ID to the ADMIN responsibility. Then give Sue a security attribute CUSTOMER_ID value of 1000.

When Sue logs into the Admin responsibility the only customer data she will have access to will have a CUSTOMER_ID value of 1000.

Note: Users can have multiple values made available to them.

See: Users Window, *Oracle Applications User's Guide, Release 11* and Responsibilities Window, *Oracle Applications User's Guide, Release 11*.

Excluding Attributes for Column–Level Security

Excluding attributes prevent certain columns of data from being visible to specified responsibilities.

For example, if for security reasons you did not want the hypothetical user Sue in the ADMIN responsibility to see data in the CONTACT_NAME column, you would assign her the excluding attribute CONTACT_NAME to the ADMIN responsibility. No users with the ADMIN responsibility can see CONTACT_NAME information.

See: Responsibilities Window, *Oracle Applications User's Guide, Release 11*.

Seeded Securing Attributes

Assign a securing attribute and value to define an attribute that must be matched by the user to see records. Attributes are defined using the Web Applications Dictionary. Assign securing attribute values for each user, and for each securing attribute assigned to all responsibilities for this user.

You may designate a user as an employee, supplier, and / or customer. This automatically assigns a contact ID value to this user for appropriate securing attributes as follows:

Contact	ID
Customer Contact	ICX_CUSTOMER_CONTACT_ID
Internal Contact	ICX_HR_PERSON_ID
Supplier Contact	ICX_SUPPLIER_CONTACT_ID

In addition, the following securing attributes are seeded:

Contact	ID
Customer	ICX_CUSTOMER_ORG_ID
Organization	ICX_HRG_ORG_ID
Supplier	ICX_SUPPLIER_ORG_ID
Customer Site	ICX_CUSTOMER_SITE_ID
Internal Site (location)	ICX_HR_SITE_ID
Supplier Site	ICX_SUPPLIER_SITE_ID

Predefined Security at Responsibility Level

The following list shows which responsibilities have predefined securing and excluding attributes:

Responsibility	Securing Attributes	Excluding Attributes
Credit Cards	ICX_HR_PERSON_ID	
Customer Registration		
Customer Services (Full Access)		
Customer Services (by Customer)		
Customer Services (by Customer Contact)		
EDI Transmissions (by Customer Site)	ICX_CUSTOMER_SITE_ID	
EDI Transmissions (Full Access)		
Events and Seminars		
Executive Overview		
Expense Reports		
Expense Reporting		
Global Assets Information		
Partner Information (by Customer)	ICX_CUSTOMER_ORG_ID	
Payments and Credits (by Customer)	ICX_CUSTOMER_ORG_ID	

Table 1 – 1 (Page 1 of 2)

Responsibility	Securing Attributes	Excluding Attributes
Payments and Credits (Full Access)		
Plan Inquiries		
Products and Orders (by Customer Contact)	ICX_CUSTOMER_CONTACT_ID	
Products and Orders (Full Access)		
Products and Orders (Guest Access)		
Project Control (by Employee)	ICX_HR_PERSON_ID	
Project Information (by Customer)	ICX_CUSTOMER_ORG_ID	
Purchasing		
Registration		
Requisitions		
Requisitions (by Preparer)	PREPARER_ID	
Requisitions (by Requester)	ICX_REQUESTOR_ID	
Requisitions (Full Access)		
Salesperson Services (by Employee)	ICX_CUSTOMER_ORG_ID	
Salesperson Services (Full Access)		
Service and Support (Full Access)		CS_PUBLIC_COMMENT
Service and Support (by Customer Contact)	ICX_CUSTOMER_CONTACT_ID	CS_COMMENT
Service and Support (by Customer)	ICX_CUSTOMER_ORG_ID	CS_COMMENT
Supplier Registration		
Supplier Services	ICX_LEVEL_ALTERED	ICX_DISTRIBUTION_ID, ICX_SUPPLIER, ICX_SUPSITE
Supplier Services (by Supplier Site)	ICX_LEVEL_ALTERED, ICX_SUPPLIER_SITE_ID	ICX_DISTRIBUTION_ID, ICX_SUPPLIER, ICX_SUPSITE
Supplier Services (by Supplier)	ICX_LEVEL_ALTERED, ICX_SUPPLIER_ORG_ID	ICX_DISTRIBUTION_ID, ICX_SUPPLIER, ICX_SUPSITE
Supplier Services (Full Access)		ICX_DISTRIBUTION_ID
Web Planning Inquiries		

Table 1 – 1 (Page 2 of 2)

Query Processing

When a user queries for data using Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers, the Web Applications Dictionary determines if any securing attributes exist in a region, and, if so, determines whether the securing attributes match those assigned to the responsibility.

If there are securing attributes assigned at the responsibility level that exactly match those at the region level, securing attribute values are checked at the user level.

If there are no securing attributes assigned at the user level that match, no data is returned. If there are securing attributes assigned at the user level that match, data is returned to the user, but only if the *user's* securing attribute values exactly match the values of the returned data.

Excluded attributes assigned at the responsibility level prevent data being returned for these attributes.

For example, assume that Sue has the following attribute values:

Securing Attribute	**Value**
CUSTOMER_ID	1000
SITE_ID	123
SITE_ID	345
SITE_ID	567
CONTACT_ID	9876

Table 1 – 2

Assume that Sue requests data for CUSTOMER_ID, SITE_ID, or CONTACT_ID, and these attributes are defined in Web Applications Dictionary and for the Customer responsibility. For any rows of data with these attributes, Sue's securing attribute values are checked for exact matches.

In this case, any rows with a CUSTOMER_ID of 1000; SITE_ID of 123, 345, or 567; and CONTACT_ID of 9876 are returned.

See Also

Web Applications Dictionary: page 3 – 2

Users Window, *Oracle Applications User's Guide, Release 11*

Responsibilities Window, *Oracle Applications User's Guide, Release 11*

CHAPTER

2

Implementation

This chapter informs you how to implement Oracle Self–Service Web Applications:

- Changing the System Administrator Password: page 2 – 7
- Setting Up Oracle Self–Service Web Applications: page 2 – 2
- Setting Up Oracle Web Application Server 3.0: page 2 – 3
- Administering Oracle Applications Security: page 2 – 4
- Customizing Your Web Pages: page 2 – 10
- Optional Setup Tasks: page 2 – 12
- Profile Options: page 2 – 16

These tasks are performed using a web browser interface. There are additional implementation tasks (for most users) for which you must use the Web Applications Dictionary. The Web Applications Dictionary is based on the Network Computing Architecture (NCA) interface. For further information, see the next chapter: Web Applications Dictionary: page 3 – 1.

Note: There may be additional setup information specific to Web Customers, Web Employees, and Web Suppliers, or features thereof. See your online HTML documentation for further product– or feature–specific setup information.

Setting Up

You must set up employee, customer and supplier records in Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers, respectively. Employee records already in Oracle Human Resources can be set up easily. Customer and supplier records from Oracle Order Entry and Oracle Purchasing, respectively, must also be added.

Prerequisite Setup Steps in Oracle Applications

❑ Set up your profile options. See: Profile Options: page 2 – 16.

❑ Register users as employees, customers and / or suppliers.

Be sure to register your user as either a employee, customer, or supplier: See: Users Window, *Oracle Applications System Administrator's Guide, Release 11*.

Employee: This user must be a valid employee in your Human Resources (HR) tables. See: Enter Person, *Oracle Human Resources User's Guide, Release 11*.

Customer: This user must exist as a customer contact in your Receivables (RA) tables. Entering Customers, *Oracle Receivables User's Guide, Release 11*.

Supplier: This user must exist as a supplier contact within Purchasing (PO) tables. See: Defining Supplier Lists, *Oracle Purchasing User's Guide, Release 11*.

❑ Assign responsibilities to users.

Once you have created responsibilities, you must assign them to individual users. You must also assign securing attribute values to users. See: Users Window, *Oracle Applications System Administrator's Guide, Release 11* and Data Security: page 1 – 15.

Setting Up Oracle Web Application Server 3.0

The following procedure must be performed to configure Oracle Application Server 3.0 for Oracle Applications Release 11. See: *Oracle Web Application Server 3.0 Installation Guide*, and other online documentation for Web Application Server 3.0.

▶ **To set up Web Application Server:**

1. Install JDBC (Java Database Connectivity) Version 1.0.2, available from the Oracle8 V8.0.4 CD. This must be installed on the same tier and in the same ORACLE_HOME directory as the Web Application Server.

2. Set up the Web Application Server Java Cartridge to use the new JDBC drivers. Use your browser to navigate to the Oracle Web Application Server Setup Page (http://<*webserver*>*:* <*admin port*>) > Web Application Server Manager > Oracle Web Application Server > Cartridge Administration > Java Cartridge specific parameters.

   Add the full path name of the JDBC classes102.zip file to the CLASSPATH environment variable. Then add the full path name of the JDBC /lib directory to the LD_LIBRARY_PATH variable.

3. Set up the Oracle Web Application Server Java Cartridge virtual paths. Use your browser to navigate to the Oracle Web Application Server Setup Page (http://<*webserver*>*:* <*admin port*>) > Web Application Server Manager > Oracle Web Application Server > Cartridge Administration > Java Cartridge > Web Request Broker parameters for Java.

   Set up the virtual path /OA_JAVA_SERV to be the same as $JAVA_TOP you set up with autoinstall.

4. Set up the Oracle Web Application Server virtual paths. Use your browser to Navigate to the Oracle Web Application Server Setup Page (http://<*webserver*>*:* <*admin port*>) > Web Application Server Manager > Oracle Web Listener > Configure (for the listener you are running for Oracle Applications) > Directory.

   In the Directory Mappings section, set up the following parameters:

```
$JAVA_TOP                        NR      /OA_JAVA/
$<doc path>                      NR      /OA_DOC/
$HTML_TOP                        NR      /OA_HTML/
$JAVA_TOP/oracle/apps/media      NR      /OA_MEDIA/
HTML_TOP/bin                     CN      /OA_HTML/bin/
```

where $JAVA_TOP and $HTML_TOP are determined via autoinstall, and `doc path` is the area unloaded from the Documentation CD.

Note: All virtual directories for Web Application Server are in upper case type and contain underscores (_).

All CGIs belong in the OA_HTML/bin directory. These should be converted to Web Application Server cartridges as soon as possible.

The OA_MEDIA directory is a subdirectory of OA_JAVA. This is necessary to allow the creation of .jar files containing graphic images.

Administering Oracle Applications Security

Because Release 11 is deployed in a multi–tier configuration, the security model has been enhanced to include authentication of application servers to the database servers they access. When this layer of security is activated, it uses ”server IDs” or passwords that the application server passes to the database server. If the database server recognizes the server ID, it grants access to the database. The server IDs are created using a Java script.

The application server security system is initially not activated; you have activate it after installation. The application servers are not assigned server IDs and the database servers do not check for server IDs.

The Java script AdminAppServer is used to set up, activate, and check the status of the application server security feature. The syntax for the script begins with the call to the script:

```
java oracle.apps.fnd.security.AdminAppServer [parameters]
```

The first parameter must be the connect string followed by the command.

```
apps/apps@dbname
ADD
```

Some commands require additional parameter(s). For example, the ADD command must be followed by the GWYUID and FNDNAM parameters, which are followed by any optional parameters. Optional parameters are indicated with an asterisk (*) next to the command.

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
   ADD                                                          \
   GWYUID=pub/pub FNDNAM=apps                                   \
*   SERVER_ADDRESS=<tcp.ip address>                            \
*   SERVER_DESCRIPTION=”Public web access server”              \
*   <env_name>=<env_value>                                     \
*   SECURE_PATH=$FND_TOP/secure
    GUEST_USER_PWD=<username/password>
```

In the example above, the parameter `env_name` allows you to enter additional information you wish to store with the server ID.

The optional parameter SECURE_PATH can be used in cases where the web server is unavailable, but the location should always be $FNDTOP/secure.

Creating Server IDs

Use the AdminAppServer script to create a server ID for the application server to access the database server. To access additional database servers from the same application server, you must rerun the AdminAppServer script for each additional database. You must run the AdminAppServer script each time you create a server ID, and each server ID only allows access to one database.

☞ **Attention:** To run the AdminAppServer script you must include $JAVA_TOP in your CLASSPATH environment variable (registry variable in Windows NT) for the application server.

▶ **To create a server ID for an application server:**

■ From the command line, enter:

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
    ADD                                                         \
    GWYUID=pub/pub FNDNAM=apps                                  \
*   SERVER_ADDRESS=<tcp.ip address>                            \
*   SERVER_DESCRIPTION=”Public web access server”              \
*   <env_name>=<env_value>                                     \
*   SECURE_PATH=$FND_TOP/secure                                \
    GUEST_USER_PWD=<username/password>
```

Note: Because the application server security feature is not initially active, assigning a server ID does not affect runtime behavior.

☞ **Attention:** If you have installed Oracle Self–Service Web Applications, you must set the `username/password` value for the GUEST_USER_PWD parameter. To do that, you must first create a valid username (”visitor” for example) in Oracle Applications. Then use the `username/password` combination as the value for GUEST_USER_PWD. The syntax is illustrated in the following example:

```
GUEST_USER_PWD=visitor/welcome
```

Oracle recommends that you do not assign any responsibilities for this user.

The GUEST_USER_PWD parameter is optional if you do not install Oracle Self–Service Web Applications.

Server ID Status

You can check the server ID for a particular database using the STATUS command in the AdminAppServer script. The STATUS command displays all registered application servers and their server IDs. The command also indicates the server security feature that is currently active.

☞ **Attention:** Check the server ID status of your databases before you activate server security.

▶ **To check the server ID status for a database:**

- From the command line, enter:

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
   STATUS
```

Activation of Server Security

You can turn the server security feature on or off using the same AdminAppServer script. When you turn off server security, you will not change or delete your server IDs. You can restart server security without recreating server IDs for all of your applications servers.

▶ **To activate server security:**

- From the command line, enter:

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
   AUTHENTICATION ON
```

▶ **To deactivate server security:**

- From the command line, enter:

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
   AUTHENTICATION OFF
```

Updating or Deleting Server IDs

You can update or delete a application server's server ID at any time. When updating the server ID you can change as many parameters as you want, including the server ID, but you must enter at least one.

▶ **To update a server ID:**

- From the command line, enter:

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
    UPDATE                                                      \
*   SERVER_ID                                                   \
*   SERVER_ADDRESS=<tcp.ip address>                             \
*   SERVER_DESCRIPTION="Public web access server"               \
*   <env_name>=<env_value>                                      \
*   SECURE_PATH=$FND_TOP/secure                                 \
    GUEST_USER_PWD=<username/password>
```

☞ **Attention:** If you have not already set the `username/password` value for the GUEST_USER_PWD parameter, you can do so here using the UPDATE command. For instructions, see the section above on Creating Server IDs.

▶ **To delete a server ID:**

- From the command line, enter:

```
java oracle.apps.fnd.security.AdminAppServer apps/apps@dbname   \
    DELETE                                                      \
    SERVER_ADDRESS=<tcp.ip address>                             \
*   SECURE_PATH=$FND_TOP/secure
```

Changing the System Administrator Password

The system administrator password for Oracle Self–Service Web Applications is the same as that for Oracle Applications. When you change a password in Oracle Applications, you are also changing it in Oracle Self–Service Web Applications, and vice versa.

Usually, the system administrator password (for the SYSADMIN user ID), is changed soon after the installation of Oracle Applications. (The predefined default password is SYSADMIN.) If the system administrator password has already been changed, you do not need to read this section.

If not, you can change it in Oracle Self–Service Web Applications.

▶ **To change the system administrator password:**

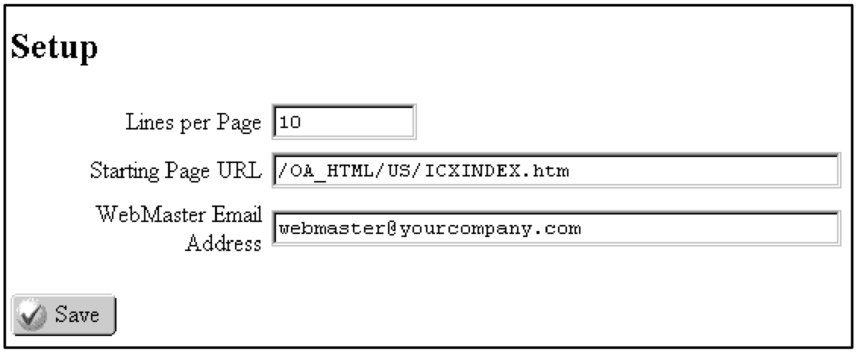
1. If the SYSADMIN password has not already been changed in Oracle Applications, continue with the next step. Otherwise, do not continue.
2. Log in to Oracle Self–Service Web Applications.
3. From the Welcome page, click General Preferences and change your SYSADMIN password. (Case is irrelevant.)

Setting Up Oracle Self–Service Web Applications

In addition to setting up common functions across all of Oracle Self–Service Web Applications, there are additional setup steps for each product.

▶ **To set up general application options:**

1. From the Welcome page, click General Application Options to open the Setup page.


Setup

Lines per Page	10
Starting Page URL	/OA_HTML/US/ICXINDEX.htm
WebMaster Email Address	webmaster@yourcompany.com

2. Enter the number of lines per page.

   This is the maximum number of table rows displayed on a page. For large tables generally, performance improves with fewer lines per page.

   If the number of rows returned exceeds the lines per page value, tables are displayed in sets.

3. Enter the starting page URL. Oracle recommends that you set this to be your login page. This is the first page users see when logging in and the page that appears after logging out.
4. Enter your webmaster's email address, where all users should send questions and comments.
5. Click Save.

Deleting Data from Temporary Tables:

Data from Oracle Self–Service Web Application's temporary tables must be deleted on a regular basis. If you do not regularly delete temporary data, temporary tables keep growing. Oracle recommends that you set up the following programs to run on a regular schedule.

> **Note:** You must perform this step even if you do not install Oracle Self–Service Web Applications. Some functions of the Self–Service Web Applications are available to the main Oracle Applications. If those functions are used, the Self–Service Web Applications temporary tables continue to grow.

▶ To delete data in temporary tables:

1. Using the Self–Service Web Applications responsibility in Oracle Applications, navigate to the Submit Request window.
2. When prompted, select Single Request.
3. Choose the list of values icon and select Delete Data from Temporary Tables.
4. Enter scheduling options. For best performance, set up this program to run on a regular basis several times per day, for example, every 30 minutes.

▶ To delete temporary data of purchase order (PO) revisions:

1. Using the Self–Service Web Applications responsibility in Oracle Applications, navigate to the Submit Request window.
2. When prompted, select Single Request.
3. Choose the list of values icon and select Delete Temporary Data of PO Revisions.
4. In the Parameters window, enter a date prior to which you want data deleted. Choose OK in the Parameters window.

5. Enter scheduling options. For best performance, set up this program to run on a periodic basis several times per day, for example, every 30 minutes.

See Also

Submitting a Request, *Oracle Applications User's Guide, Release 11*

Customizing Your Web Pages

The following steps explain how to customize certain aspects of your Oracle Self–Service Web Applications pages.

▶ **To add your company logo:**

You can replace the default Oracle logo with your own corporate logo. Your logo will then appear on every page.

1. Create a GIF file containing your corporate logo and name it FNDLOGOS.gif.
2. Place the file in the `<OA_MEDIA>` directory as defined in the Web Listener.

   If you have a multilingual install, you must also copy this file into the other language location.

▶ **To change the background color:**

You can replace the background on every page with your own choice of background color and texture.

1. Create a JPEG file containing your background and name it ICXBCKGR.jpg.
2. Place the file in the `<OA_MEDIA>` directory as defined in the Web Listener.

   If you have a multilingual install, you must also copy this file into the other language location.

▶ **To customize the Universal Home Page:**

The Universal Desktop home page consists of two frames, each referencing its own HTML document. You may customize any of these frames as follows:

1. For the left frame, edit the file `<OA_MEDIA>` directory as defined in the Web Listener. You can replace the existing news items and links and/or add new items.
2. For the right frame, edit the file `<`*`path`*`>/ic_source/html/US/ICXUDUL.htm` You can replace the existing icons or links or add new ones.

▶ **To link Self–Service Web Applications to your personal home page:**

- You can provide a login prompt on your own home page and access Oracle Self–Service Web Applications using the following URL:

  `http://<host:port>/<DCDname>/owa/OracleApps.DisplayLogin`

Optional Setup Tasks

These setup steps depend upon which Self–Service Web Applications, (and features) you have installed.

Web Employees (Requisitions)

Requisition Template Hierarchies

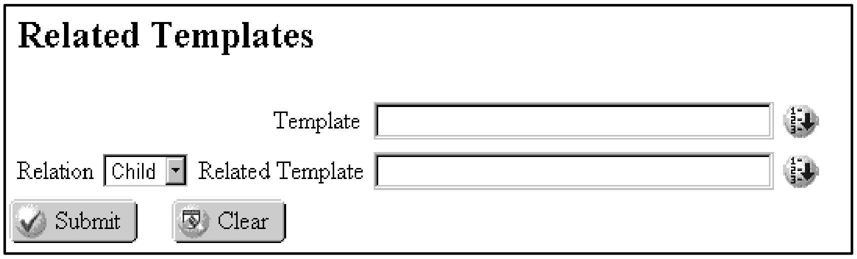
This page provides setup for Web Requisition template hierarchies. You may establish template hierarchies for your requisition function by assigning parent and child relationships.

For example, suppose you are designing a requisition order system and want to offer all office furniture under the template ”Office Furniture.” You have several subordinate templates such as Desks, Chairs, Bookcases, and Computers. You will need to set up one ”Top” template (Office Furniture, leaving Related Template blank), and then relate each furniture template type as ”Child” template to that top template. Each of furniture type is a ”Related Template.” The result of setting up this hierarchy is then:

- Office Furniture
  - Desks
  - Chairs
  - Bookcases
  - Computers

▶ **To set up a hierarchy:**

1. Enter the template name, or select a template from the list of values.



Related Templates

Template

Relation Child Related Template

2. Select a relation for the template. Select Top if this is an initial or top level template (Furniture, for example). Select Child to make

this a parent template for the one you will enter for the Related Template.

3. Enter a related template (Desks, for example), or the next subordinate template, if there is one.
4. Click Submit.

   If the child level template has templates of its own, continue the structure by entering the child level template and then enter the related subordinate template.

Web Customers (Store) and Web Employees (Requisitions)

Item Category Hierarchies

This page provides setup information for the Web Store and Web Requisition Category hierarchies. You may establish category hierarchies for your own Web Store or your Requisition function, by assigning parent and child category relationships.

For example, suppose you are designing a requisition order system and want to offer office furniture under the category ”Office Furniture.” Assume you have several subordinate categories such as Desks, Chairs, Bookcases, and Computers. You will need to set up one ”Top” category and then relate each furniture category type as ”Child” categories to that ”Top” category. Each of the furniture types is a ”Related Category.” The result of setting up this hierarchy is:

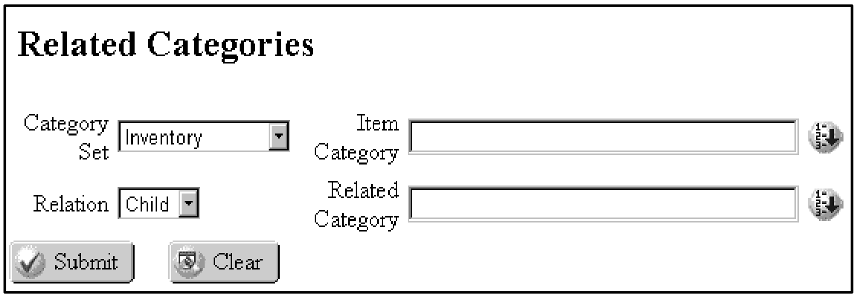
- Office Furniture
  - Desks
  - Chairs
  - Bookcases
  - Computers

Prerequisites

❑ Create category sets in Oracle Inventory.

▶ **To set up a hierarchy:**

1. Select a category set from list of values.



2. Select an item category from the list of values.
3. Select a relation for the category. Select Top if this is an initial or top level category (Furniture, for example). Select Child to make this a parent category for the one you will enter for the Related Category.
4. Select a related category (Desks, for example), or the next subordinate category, if there is one.
5. Click Submit.

   If the Child level category has subcategories of its own, continue the structure by entering the Child level category in the Category field, then enter the subordinate category in the Related Category field.

Web Employees (Expenses)

The Global Assets Registry queries descriptive and financial information about your assets system.

Prerequisites

1. Install Oracle Assets and Payables
2. Complete all setup steps for the Global Assets Registry.
3. Verify that the following key Asset flexfields have been compiled using the Key Flexfield Segments window.
   - Accounting flexfield
   - Asset Category flexfield

- Location flexfield
- Asset Key flexfield

See: *Oracle Assets User's Guide, Release 11* and *Oracle Applications Flexfields Guide, Release 11*.

Web Employees (Requisitions) and Web Suppliers

▶ **To set up links to supplier items:**

■ Use the ATTRIBUTE14 descriptive flexfield segment in the PO_LINES_ALL table to specify an additional URL to support drilldown to your freight carriers. For each supplier item for which you support drilldown, enter the static URL for the carrier–specific web page in ATTRIBUTE14.

Use the ATTRIBUTE14 descriptive flexfield segment in the PO_VENDORS table to specify an additional URL to support drilldown to your suppliers. For each supplier for which you support drilldown, enter the static URL for the carrier–specific web page in ATTRIBUTE14.

See: *Oracle Applications Flexfields Guide, Release 11*.

Web Customers (Store)

▶ **To set up freight carrier links.**

■ Use the ATTRIBUTE14 descriptive flexfield segment in the ORG_FREIGHT table to specify an additional URL to support drilldown to your freight carriers. For each freight carrier for which you support drilldown, enter the static URL for the carrier–specific web page in ATTRIBUTE14.

▶ **To set up links to items:**

■ Use the ATTRIBUTE14 descriptive flexfield segment in the MTL_SYSTEM_ITEMS table to specify an additional URL to support drilldown to your items. For each item for which you support drilldown, enter the static URL for the carrier–specific web page in ATTRIBUTE14.

See: *Oracle Applications Flexfields Guide, Release 11*.

Profile Options

During implementation, the system administrator sets up and maintains profile options.

CZ: Use Simple Configurator

A value of Yes (default) indicates that the simple, HTML version (no frame support) of the Web Configurator is used. A value of No indicates that the Java version (supports frames) is used.

FND: Applications Web Agent

Provides the base URL for the Apps Schema's Application Server DAD. Your System Administrator sets this profile option during the install process. The syntax takes the form:

```
http://<application server machine name>/<DAD name>/
```

ICX: Allow Funds Override

If encumbrance is enabled, indicates whether a requestor can override their allowed funds.

ICX: Date Format Mask

Determines the date format mask to use. The American English default is DD–MON–RRRR, for example, 12–NOV–1997.

For year 2000 compliance, all year formats are converted to RRRR, which accepts four–digit century and year entries verbatim (1950 is stored as 1950) and converts two–digit year entries as follows:

- Entries of 00 to 49 are converted to 2000 to 2049, respectively.
- Entries of 50 to 99 are converted to 1950 to 1999, respectively.

For example, if a user enters 50 for the year, the year is converted and stored as 1950. If a user enters 49, the year is converted and stored as 2049.

ICX: Days Needed By

Determines the number of days until the user needs the order.

ICX: Default Employee

Determines the default employee to use.

ICX: Default Requisition Template

Determines the default requisition template to use.

ICX: Language

Determines the default language.

ICX: Limit Connect

Determines the maximum number of page hits per session.

ICX: Limit Time

Determines the maximum number of hours a user can be logged on per session.

ICX: Override Location Flag

Determines whether the default location to deliver orders can be overridden.

ICX: Override Requestor Code

Determines whether the user can override the default requestor code and create a requisition for everyone, the entire organization, or for just the user.

Key	
✓	You can update the profile option.
–	You can view the profile option value but you cannot change it.

Profile Option	User Access	System Administrator				Requirements
		User	Resp	App	Site	Default Value
CZ: Use Simple Configurator					✓	Yes
FND: Applications Web Agent	–	✓	✓	✓	✓	
ICX: Allow Funds Override	–	✓	–	–	–	
ICX: Date Format Mask	✓	✓			✓	DD–MON–RRRR. For example, 08–MAR–1998.
ICX: Days Needed By	✓	✓	✓	✓	✓	2
ICX: Default Employee	–	✓	✓	✓	✓	

Profile Option	User Access	System Administrator				Requirements
ICX: Default Requisition Template	✓	✓	✓	✓	✓	
ICX: Language	✓	✓			✓	American English
ICX: Limit Connect	–	✓			✓	1000
ICX: Limit Time	–	✓			✓	4
ICX: Override Location Flag		✓	✓	✓	✓	Yes
ICX: Override Requestor Code		✓	✓	✓	✓	No

CHAPTER

3

Web Applications Dictionary

This chapter discusses the Web Applications Dictionary, the data repository for Oracle Self–Service Web Applications. While Web Applications Dictionary is not absolutely necessary for your implementation process, it *is* necessary if you customize.

Web Applications Dictionary Overview

The Web Application Dictionary is an active data dictionary that enables you to define inquiry applications for the web, and generate many of the application's characteristics at runtime. The data dictionary stores key information about your application, including appearance, language, security requirements, navigation, and data. Because this information is stored in an active data dictionary, you can create an inquiry application for the web specifically designed to meet your business needs.

An Oracle Forms user–interface is provided for you to enter your application's characteristics in the active data dictionary. Through this user–interface, you can customize existing inquiry applications for the web, or create new ones without programming effort. You can create applications that are customizable, extensible, and multi–lingual.

With Oracle Web Application Dictionary you can:

- Develop inquiry applications for the web without programming
- Generate the inquiry application web pages at runtime
- Register your application definition in an active data dictionary
- Customize and extend existing applications, and maintain your customizations
- Seamlessly integrate Oracle Applications data and company intranet content
- Completely reconcile company transactions through a web inquiry interface
- Graphically illustrate your application data relationships using Object Navigator

Definitions

Object

A database view.

Attribute

A reusable field used in a web inquiry application. For example, customer name and customer number are both attributes. An attribute is not associated with data. For example, the customer name attribute can be reused anytime a customer name field is displayed on a web inquiry screen.

Object Attribute

A reusable field that results when you associate an attribute with an object.

Flow

An illustration of data relationships. A flow may be exhibited in the form of a series of web pages, each displaying data and its relationship to other data. A flow may also assume a hierarchical representation in the Object Navigator.

Page (or Flow Page)

A page as defined in the Web Application Dictionary becomes a web page in the flow of your application.

Region

A logical grouping of data. For example, customer information can be grouped in one region and shipping information can be grouped in another region. A region also represents a section of a web page.

Page Region

A region associated with a page.

Primary Region

The first region of a page.

Region Item

A reusable field that results when you associate an attribute or object attribute with a region.

Designing a Web Inquiry Application

Before actually registering your application in Web Application Dictionary, you must design not only the look and feel of the application, but also the supporting logical data model. You must identify the database tables that store the data to be displayed in your web inquiry application.

Because the Web Application Dictionary derives its data from database views, you must create views on the relevant database tables. You can join multiple tables to create a view, or simply create a view for each table.

This preparation is essential to your success in creating a web inquiry application.

Creating a Flow

Use the Web Application Dictionary to create flows.

Flow Components

The components of a flow are:

- Objects
- Attributes
- Object Attributes
- Pages
- Regions
- Region Items
- Page Regions
- Links

Objects

You must create one (and only one) object for each of your database views.

Attributes

Both objects and attributes comprise the backbone of a flow. You can reuse them in many flows.

You must create an attribute for each column of your database view. For example, suppose you have a view on the CUSTOMER table and the view contains the columns CUSTOMER_ID and CUSTOMER_NAME. You must create an attribute for both, even though you may not want to display the CUSTOMER_ID. When you create an attribute, you can indicate various display options, including Hidden.

The attribute definition serves as the basis of your subsequent object attribute and region item definitions.

Although you create an attribute for each view column, the attribute itself is not associated with a database column, and hence is not associated with data.

Object Attributes

You may reuse attributes in many flows. When you create an *object attribute*, you are restricting the attribute definition to a particular object. For example, once you associate the CUSTOMER_NAME attribute with the CUSTOMER view, you have limited the definition of CUSTOMER_NAME to its corresponding column in the CUSTOMER view. You do not, however, lose your original *attribute* definition. This is maintained, and may be continually reused.

The characteristics of an *object attribute* are inherited from the original *attribute* definition. You may override these defaulted characteristics. Any characteristics you override only apply to the *object attribute* definition; the original *attribute* definition remains unaffected.

Object attributes are associated with data in the database. Therefore, to display data for a particular field on a web page, you must create an object attribute for that field.

Pages

You must register each of your web pages in the Web Application Dictionary. For example, if you want one web page to display the customer name and number, and another web page to display the customer address, you must register two pages in the Web Application Dictionary.

> **Note:** Pages are not reusable. A page only exists within the context of its flow.

Regions

A Region is simply a section of a web page. Suppose, for example, that you want a web page to display both the customer name, number, and the address. You would likely want this information illustrated in two separate sections on the same web page. This design would require that you define two regions in the Web Application Dictionary.

Each region is based upon *one and only one* object. The Web Application Dictionary determines the data to display in a region from the region's underlying object.

Region Items

You must define a region item for each field you want to display in a region. In the example above, you would define six region items, one for each displayed field: customer name, number, address, city, state, and zip code. Region items typically represent only those fields that you want to *display* in the region.

The region item definition is defaulted from the original object attribute definition, although you may override the defaults. Any overridden defaults only apply to the region item definition; the original object attribute definition is not affected.

Page Regions

Like attributes, you can reuse regions in many flows. To specify that a particular page contains a region, you must create a page region.

Links

Using the Web Application Dictionary, you can define hypertext links between the web pages in your inquiry application.

You can define a hypertext link to an external web site as well. To do this, you must define an object attribute of datatype URL. This object attribute serves as a placeholder for the external URL address. You must then place the URL attribute in the region containing the hypertext link (using the region items window).

Steps to Creating a Flow

Step	**Window / Navigation**
	Text in brackets ([]) indicates a button.
Design the flow	Not applicable. Create a navigation plan and database views.
Define an object	Objects window / Navigator > Object Workbench. See Defining Objects: page 3 – 22.
Define attributes for the object	Attributes window / Navigator > Object Workbench > [Create Attributes]. Choose the Create Attributes button immediately upon opening the Object Attributes window. See: Defining Attributes: page 3 – 25.
Add attributes to the object to create object attributes	Object Attributes window / Navigator > Object Workbench. Close the Attributes window to return to the Object window. See Assigning Attributes to Objects: page 3 – 23.
Repeat the three steps above for each object.	
Define primary keys for each object	Unique Keys window / Navigator > Object Workbench > [Primary Keys]. Select an object in the Objects window and choose the Primary Keys button. See: Defining Primary Keys: page 3 – 26.

Table 3 – 1 Steps to Creating a Flow

Step	Window / Navigation
Define foreign keys for each object	Foreign Keys window / Navigator > Object Workbench > [Foreign Keys]. Select an object in the Objects window and choose the Foreign Keys button. See: Defining Foreign Keys: page 3 – 27.
Identify primary unique key for each object	Objects window / Navigator > Object Workbench. Close the Foreign Keys window to return to the Objects window. See Defining Objects: page 3 – 22.
Define all regions	Regions window / Navigator > Region Workbench. See: Defining Regions: page 3 – 29.
Select a region and add attributes to it to create region items. *Repeat this step for each region.*	Region Items window / Navigator > Region Workbench > [Region Items]. See: Creating Region Items: page 3 – 31.
Define a flow name	Flows window / Navigator > Flow Workbench. See: Defining Object Flows: page 3 – 32.
Define all flow pages	Flow Pages window / Navigator > Flow Workbench > [Page Regions]. See: Defining Flow Pages: page 3 – 33.
Select a page and add regions to it to create page regions. *Repeat for each page.*	Page Regions window / Navigator > Flow Workbench > [Flow Pages] > [Page Regions]. See: Defining Flow Page Regions: page 3 – 35.
Define all page relationships	Page Relations window / Navigator > Flow Workbench > [Flow Pages] > [Page Relations]. See: Defining Flow Page Relations: page 3 – 37.
Select a page region and define its hyperlinks. *Repeat for each page region with a link.*	Links window / Navigator > Flow Workbench > [Flow Pages] > [Page Regions] > [Links]. See: Defining Flow Page Region Links: page 3 – 38.
Optionally, run the flow in Object Navigator	Run Flows window / Navigator > Flow Workbench > [Run] > [Run]. See: Defining Object Flows: page 3 – 32.

Table 3 – 1 Steps to Creating a Flow

Note: There are alternative ways of creating flows in the Web Application Dictionary. This series of steps illustrates only one option.

Example

This section uses an example to illustrate the steps involved in creating a flow. The steps listed above are described again here, but in more detail.

Step 1. Design your flow.

Before entering data into the Web Application Dictionary, you must design your web inquiry application. This involves determining the business needs you want to satisfy with the application, identifying the source of the data to be displayed, and designing the look and feel of the application.

It is suggested that you create a navigation map before actually entering data. The navigation map should include the significant aspects of your flow:

- web pages
- regions for each web page
- attributes displayed and hidden in each region (including buttons)
- objects behind each region
- views the objects are based upon
- primary key(s) for each object
- foreign key(s) for each object, if applicable
- navigation path(s) through the web pages, including hypertext links
- for each navigation path, the relationship between the From and To objects.

The list below illustrates the navigation map for this example.

For this example, the following views must be created in the database:

SO_HEADERS

Create or replace view SO_HEADER_EXAMPLE_V as

```
select
sh.header_id,
sh.order_number,
rc.customer_id,
rc.customer_name,
rc.customer_number
from so_headers sh,
ra_customers rc
where
sh.customer_id = rc.customer_id;
```

SO_LINES

Create or replace view so_lines as

```
select
sl.line_id,
sl.header_id,
sl.line_number,
sl.inventory_item_id,
sl.warehouse_id,
msi.organization_id,
msi.description item_name
from
so_lines sl,
mtl_system_items msi
where
sl.inventory_item_id =
msi.inventory_item_id and
sl.warehouse_id =
msi.organization_id;
```

Step 2. Define an object

Once you have prepared a navigation map and created views for your inquiry web application, you must register the views as objects. In this particular example, two objects are created, one at a time:

- OBJECT_SO_HEADERS (based on the view, SO_HEADERS)
- OBJECT_SO_LINES (based on the view, SO_LINES)

Step 3. Define attributes for the object

After you create an object, you can define the attributes that correspond to the object. Your attribute definitions do not apply to a particular object at this point. You must still associate your attributes to an object.

In the example, the following attributes must be created for the object, OBJECT_SO_HEADERS:

- header ID
- customer ID
- order number

- customer name
- customer number
- URL attribute (for the external shipping supplier web site)

The following attributes must be created for the object, OBJECT_SO_LINES:

- line ID
- header ID (reuse the header_id attribute defined for the object, OBJECT_SO_HEADERS)
- line number
- item

You must also create an attribute for the shipments button.

In the attributes window, you can define the following information about each of the attributes:

Attribute Information	**Field Prompt**	**Required?**
Owning Oracle Application	Application	Y
User–friendly attribute identifier	Attribute Id	Y
User–friendly attribute name (used in LOVs in later Web Application Dictionary screens)	Attribute Name	Y
Field prompt to be displayed for the attribute in the Web Application	Long Label	Y
Textual appearance of the attribute value	Bold check box, Italic check box	N
Attribute alignment on the web screens	V Align, H Align	Y
Attribute datatype	Datatype	Y
Length required to display the field prompt for the attribute	Label Length	N
Length required to display the value of the attribute	Value Length	Y
Free–form text attribute description	Description	N

Table 3 – 2

Below is an example of data that may be entered in the Attributes window for the attribute, customer name:

Attribute Information	**Field Prompt / Sample Data**
Owning Oracle Application	Application = Oracle Electronic Data Interchange
User–friendly attribute identifier	Attribute Id = Customer_Name
User–friendly attribute name (used in LOVs in later Web Application Dictionary screens)	Attribute Name = Customer Name
Field prompt to be displayed for the attribute in the Web Application	Long Label = Customer
Textual appearance of the attribute value	Bold check box = checked
Attribute alignment on the web screens	V Align = Top, H Align = Centered
Attribute datatype	Datatype = Number
Length required to display the field prompt for the attribute	Label Length = 8
Length required to display the value of the attribute	Value Length = 30
Free–form text attribute description	Description = This attribute corresponds to the name of the customer.

Table 3 – 3 (Page 1 of 1)

Step 4. Add attributes to the object to create object attributes

At this point you have defined all attributes for your web inquiry application. Now, you must associate each attribute with an object, the object containing the data to be displayed for the attribute. In this example, the following attributes are associated with the object, OBJECT_SO_HEADERS:

- header ID
- customer ID
- order number
- customer name
- customer number
- URL attribute (for the external shipping supplier web site)

The following attributes are associated with the object, OBJECT_SO_LINES:

- line ID
- header ID
- line number
- item

The data in the Object Attributes window is defaulted for you from the original attribute definitions. You may override these defaults.

Additionally, you must use the object attributes window to map each object attribute to an object database column. In this example, you would map the customer name attribute to the object column, customer_name. Use the LOV option to obtain a list of valid database columns from which to choose.

Step 5. Define primary keys for each object

You must use the Unique Keys window to define primary keys for each of your objects. For this example, you would define the following primary keys:

Object	Primary Key(s)
OBJECT_SO_HEADERS	HEADER_ID
OBJECT_SO_LINES	LINE_ID

Table 3 – 4 (Page 1 of 1)

Step 6. Define foreign keys for each object

Once you have defined all primary keys, you must use the Foreign Keys window to define the foreign keys for each of your objects. The primary key / foreign key relationships you define dictate the navigation paths through your web pages and regions. Therefore, if you intend on navigating from one region to another region based upon the same object, you must define a foreign key for that common object. In this example, you would define the following foreign keys and primary key / foreign key relationships:

Object	Foreign Key(s)	Relationship
OBJECT_SO_HEADERS	HEADER_ID	Object to Itself
OBJECT_SO_LINES	HEADER_ID	Header to Lines

Table 3 – 5 (Page 1 of 1)

Notice that the object, OBJECT_SO_HEADERS has a foreign key that is the same as the primary key, HEADER_ID. This is because this particular example requires that navigation occur from one region (Summary of Orders) to another region (Order Detail) that is based upon the same object.

Step 7. Identify the primary unique key for each object

If you define multiple unique keys for a particular object, you must use the objects window to identify the primary unique key. This is the unique key used for navigation through your web pages and regions. In this example, there is only one unique key for each object. Therefore, the primary unique key is just the one and only unique key.

Step 8. Return to Navigator window

Step 9. Define all regions

In the same way you define attributes and then associate them to objects, you must also define regions, and then associate them to pages. In this example, the Regions window is used to define the following regions:

- Summary of Orders
- Order Detail
- Order Lines

In the Regions window, you can define the following information about regions:

Region Information	Field Prompt	Required?
User–friendly region identifier	Region ID	Y
User–friendly region name (this name will be displayed at the top of the region in your web application)	Region Name	Y

Region Information	Field Prompt	Required?
Owning Oracle Application	Application Name	Y
The object underlying the region	Object Name	Y
The number of *database* rows to be displayed in the region: one or many	Region Style	Y
The number of screen columns you would like displayed in the region before the region data wraps around to the next screen line	Number of Col- umns	N
Free–form text region description	Description	N

Table 3 – 6   (Page 2 of 2)

Below is an example of data that may be entered in the Regions window for the region, Order Detail:

Region Information	Field Prompt / Sample Data
Region Information	Screen Field / Sample Data
User–friendly region identifier	Region Id = Order_Detail_Region
User–friendly region name (this name will be displayed at the top of the region in your web application)	Region Name = Order Detail
Owning Oracle Application	Application Name = Oracle Electronic Data Interchange
The object underlying the region	Object Name = object_so_headers
The number of *database* rows to be displayed in the region: one or many	Region Style = single–row
The number of screen columns you would like displayed in the region before the region data wraps around to the next screen line	Number of Columns = 4
Free–form text region description	Description = This is the Order Detail Region, used to display summary information for the drill down order

Table 3 – 7   (Page 1 of 1)

Step 10. Select a region and add attributes to it to create region items.

For each region, you must use the Region Items window to place attributes and / or object attributes in the region. Region items are typically the object attributes that you would like to *display* in the region, however, there are some exceptions to this rule: URL attributes must be defined as region items, and attributes by which you are securing data must also be defined as region items (for more information on defining attribute security, refer to the document, Web Inquiries).

In this example, you would define the following region items for the Summary of Orders region:

- order number
- customer name
- customer number

and the following region items for the Order Detail region:

- order number
- customer name
- URL attribute (for the external shipping supplier web site)

and the following region items for the Order Lines region:

- line number
- item

The data in the Region Items window is defaulted from the original attribute definitions. You may override these defaults.

In addition to the defaulted information, you can enter the following information about region items in the region items window:

Region Item Information	Field Prompt	Required?
Whether the region item is an attribute or an object attribute (a type of attribute is usually reserved for a button)	Attribute Type	Y
The order in which you would like to display the region items in the region	Display Sequence	Y
The display style of the region item. This can be one of the following styles: 1) button 2) check box 3) hidden (does not display in the region) 4) poplist 5) text	Item Style	Y
Whether a web query window should be created for the region item.	Queryable check box	N
If you would like the region item to be in an order by clause when the Web Application Dictionary selects the data from the database. And, whether you would like the order by to be ascending or descending.	Order Sequence and Order Direction	N

Table 3 – 8 (Page 1 of 1)

Step 11. Return to Navigator window

Step 12. Define a flow name

You must enter a name for your flow in the Flows window. The name of this example flow may be, View Sales Orders.

Step 13. Define all flow pages.

Once you have entered a flow name to register your flow, you must define the web pages comprising the flow. In this example, you would define the following web pages:

- Summary of Orders
- Order Detail

You can use the Flow Pages window to enter the following information about each page:

Page Information	Field Prompt	Required?
User–friendly page identifier	Page Id	Y
User–friendly page name (this name will be displayed at the top of the web page in your web application)	Page Name	Y
Free–form text page description	Description	N
Whether the page is the first page in the flow. You can only have one primary page.	Primary Page check box	Y

Table 3 – 9 (Page 1 of 1)

Step 14. Select a page and add regions to it, creating page regions.

Now that you have defined all of your web pages and regions, you can combine them to build your flow. In this example, you would use the Page Regions window to add the following region to the Summary of Orders page:

- Summary of Orders Region
- and the following regions to the to Order Detail page:
- Order Detail Region
- Order Lines Region

For each page region, the data in the Page Regions window defaults from the original region definition. You may override these defaults.

In addition, you can enter the following information about regions in the Page Regions window:

Page Region Information	Field Prompt	Required?
Whether the region is the first region of the page	Primary Region check box	Y
If the region is not the primary region, identify the region displayed before it on the web page.	Parent Region	Y
The relationship between the parent and child regions (*this relationship was initially defined in the foreign keys window*).	Relationship	Y

Table 3 – 10 (Page 1 of 1)

Step 15. Define all page relationships

You must define the relationships among all web pages. For this example you would define the following page relationships:

From Page	From Region	Target Page	Relationship (Defined in the Foreign Keys window)
Summary of Orders	Summary of Orders	Order Detail	Object to itself

Table 3 – 11 (Page 1 of 1)

Step 16. Select a page region and define its hypertext links.

If applicable, for each page region you must define hypertext links. You can specify a page or a URL attribute as the link destination. For this example, you would define a link on the order number from the Summary of Orders page (and Summary of Orders region) to the Order Detail page.

You must also define a link from the Shipments button for this example. Because the Shipments button is supposed to cause navigation to an external web site, you would define the link destination to be the URL attribute defined above.

Optional Web Application Dictionary Windows

The steps above show one way of creating a flow using the Web Application Dictionary. However, there are additional optional screens that are provided as well. They are:

Assign Regions

You can use this screen to assign an object attribute to many regions at once. You can optionally navigate to this screen from the Object Attributes window using the Multiple Assignments button.

☞ **Attention:** The Attribute Values window is not applicable to the Web Application Dictionary. The Attribute Navigation button in the Regions window causes the Attribute Values window to be displayed.

Modifying an Existing Web Inquiry Application

You can use the Web Application Dictionary to both create a new web inquiry application or modify an existing one. The table below lists some of the information you may want to modify for an existing web inquiry application and where to make the corresponding change in the Web Application Dictionary windows.

Information to Change	Where to Make the Change
Change field prompt	If you want the change made globally, **Attributes window (Long Label field)** If you want the change made for an object (and everywhere else the object is used in the flow), **Object Attributes window (Long Label field)** If you want the change made for that region (and everywhere else the region is used), **Region Items window (Long Label field)**
Extend or truncate the length of the field prompt	If you want the change made globally, **Attributes window (Label Length field)** If you want the change made for an object (and everywhere else the object is used in the flow), **Object Attributes window (Label Length field)** If you want the change made for that region (and everywhere else the region is used), **Region Items window (Label Length field)**
Extend or truncate the length of the data in a field	If you want the change made globally, **Attributes window (Value Length field)** If you want the change made for an object (and everywhere else the object is used in the flow), **Object Attributes window (Display Value Length field)** If you want the change made for that region (and everywhere else the region is used), **Region Items window (Display Length field)**

Table 3 – 12 (Page 1 of 2)

Information to Change	Where to Make the Change
Change the textual appearance of the data in a field	If you want the change made globally, **Attributes window (Bold and Italic check boxes)** If you want the change made for an object (and everywhere else the object is used in the flow), **Object Attributes window (Bold and Italic check boxes)** If you want the change made for that region (and everywhere else the region is used), **Region Items window (Bold and Italic check boxes)**
Change the alignment of a field on the screen	If you want the change made globally, **Attributes window (V Align and H Align fields)** If you want the change made for an object (and everywhere else the object is used in the flow), **Object Attributes window (V Align and H Align fields)** If you want the change made for that region (and everywhere else the region is used), **Region Items window (V Align and H Align fields)**
Change the datatype of an attribute	Attributes window (Datatype field)
Change the order by clause to include or exclude an attribute	Region Items window (Order Seq field and Order Direction field)
Generate a web query window for an attribute	Region Items window (Queryable check box)
Change the region heading	Region window (Region Name field)
Change the page heading	Flow Pages window (Page Name field)
Change the name of the flow	Flows window (Flow Name field)

Table 3 – 12 (Page 2 of 2)

Defining Objects

You must define an object for each database view to be used in your flow. This function registers the view in the Web Application Dictionary.

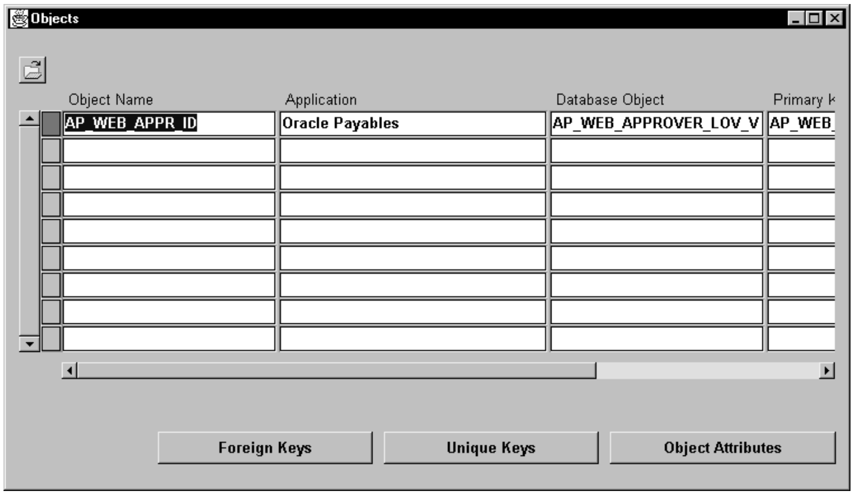
> **Note:** You can only define one object per database view.

Prerequisites

❑ Created views to use in your web inquiry.

▶ **To define an object:**

1. In the Web Application Dictionary, navigate to the Objects folder window.



Object Name	Application	Database Object	Primary K
AP_WEB_APPR_ID	Oracle Payables	AP_WEB_APPROVER_LOV_V	AP_WEB_

2. Enter an object name.
3. Select an application.
4. Select a database object, i.e., a database view.
5. Choose the Primary Keys button and define the primary and unique keys for the database object.
6. Enter the primary key.
7. Save your work.
8. Choose the Foreign Keys button to define foreign keys for the database object.

9. Choose the Object Attributes button to define the attributes for the database object.

See Also

Defining Attributes: page 3 – 25

Assigning Attributes to Objects: page 3 – 23

Defining Primary Keys: page 3 – 26

Defining Foreign Keys: page 3 – 27

Assigning Attributes to Objects

Associate defined attributes with one or more objects (database views) to create object attributes.

Note: Uniform Resource Locator (URL) attributes must be object attributes.

Note: If you are updating existing assignments and you change a long label, you are prompted if you want to change the label for all related object attributes and region items. If you choose OK, all related labels are changed.

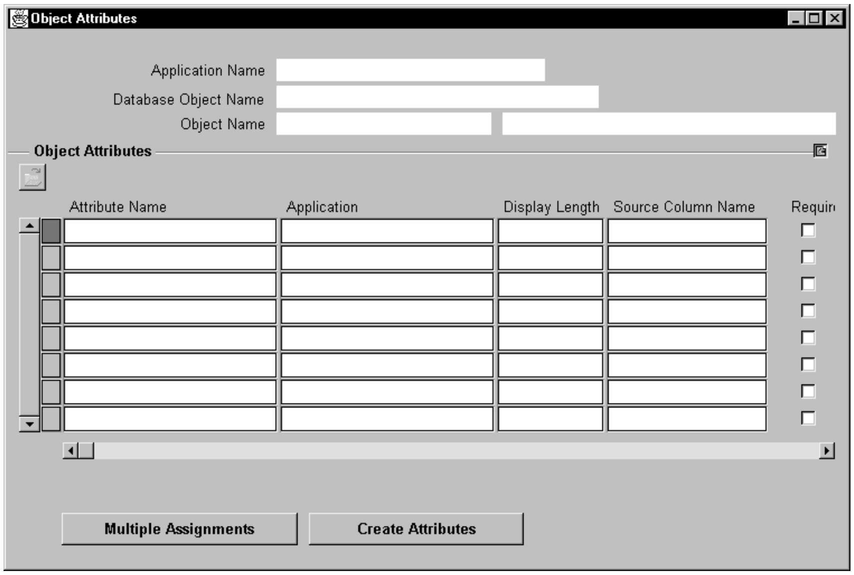
Prerequisites

❑ Define objects.

❑ Define the attributes to assign to objects.

▶ **To assign attributes to objects:**

1. Navigate to the Object Attributes folder window.



Object Attributes

Application Name	
Database Object Name	
Object Name	

Object Attributes

Attribute Name	Application	Display Length	Source Column Name	Requir
				☐
				☐
				☐
				☐
				☐
				☐
				☐
				☐

2. Select an existing attribute name to assign to an object.
3. Optionally, select a database view column name corresponding to the object attribute.
4. Enter a long label for the object attribute. The default is the label used when the attribute was defined, but can be overridden.

   Note: The remaining data in the Object Attributes folder window defaults from when you defined the attribute. You may override these defaults.
5. Choose the Create Attributes button to create additional attributes.

▶ **To assign multiple regions:**

1. Choose the Multiple Assignments button.
2. Enter the all the regions that you want the current object attribute assigned to.

See Also

Defining Objects: page 3 – 22

Defining Attributes: page 3 – 25

Defining Primary Keys: page 3 – 26

Defining Foreign Keys: page 3 – 27

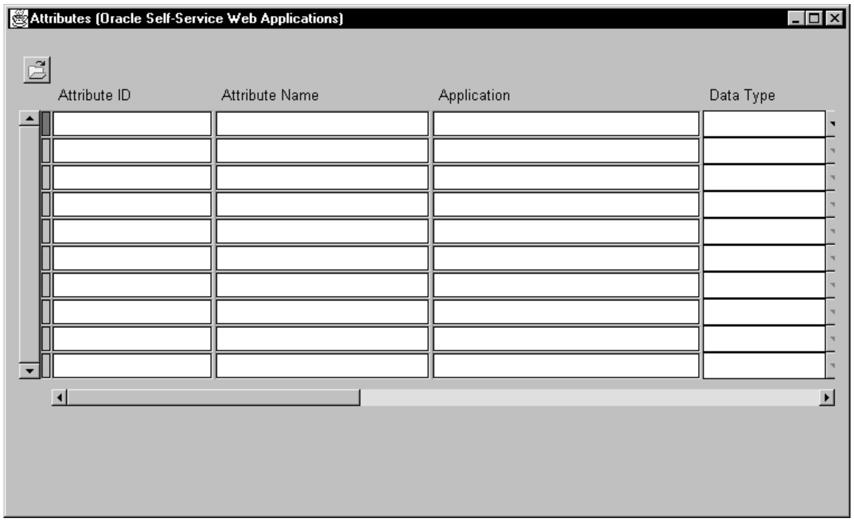
Defining Attributes

Attributes can be defined and then assigned to one or more objects.

> **Note:** If you are updating existing attributes and you change a long label, you are prompted if you want to change the label for all related object attributes and region items. If you choose OK, all related labels are changed.

▶ **To define attributes:**

1. Navigate to the Attributes folder window. Do this by choosing the Create Attributes button from the Object Attributes folder window.



2. Enter an application to associate with the attribute.
3. Enter an attribute identifier.
4. Enter a user–friendly attribute name to be used in lists of values.
5. Enter a long label for the attribute. The default is the attribute name. This is the attribute prompt in your web inquiry application.

6. Optionally, indicate how the text should appear on the browser: bold, italic, and so on.
7. Select a vertical alignment: Top, Center, or Bottom.
8. Select a horizontal alignment: Left, Center, or Right.
9. Enter the datatype for the attribute.
10. Enter the display length for the attribute value.
11. Optionally, enter a free–form description for the attribute.

See Also

Defining Objects: page 3 – 22

Assigning Attributes to Objects: page 3 – 23

Defining Primary Keys: page 3 – 26

Defining Foreign Keys: page 3 – 27

Defining Primary Keys

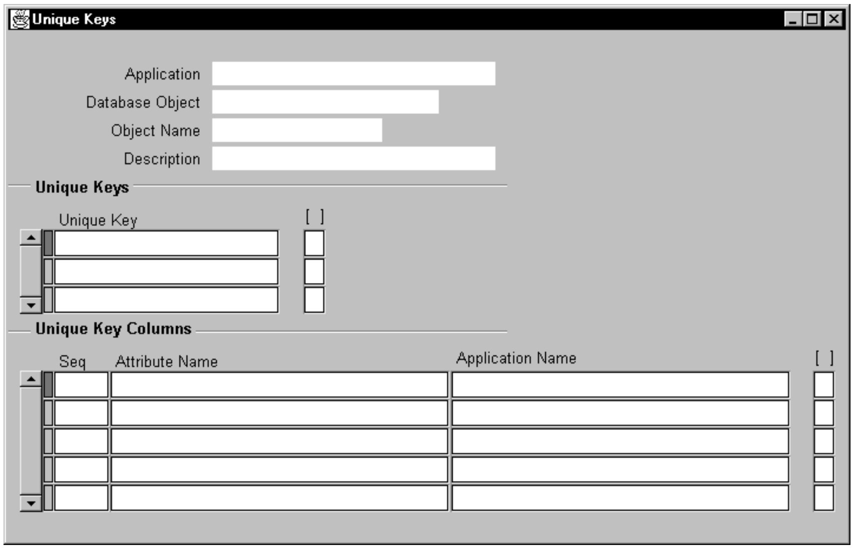
For each object, a unique primary key must be defined. A primary key ensures that each row of data can be uniquely identified and cannot be duplicated.

Prerequisites

- ❑ Define objects.
- ❑ Define attributes.
- ❑ Define object attributes.

▶ **To define a primary key:**

1. In the Web Application Dictionary, navigate to the Unique Keys window by choosing the Primary Keys button from the Objects folder window.



2. Enter a name for the unique (primary) key.

3. Enter at least one unique key column sequence. The sequence determines the order the specified columns are evaluated.

See Also

Defining Objects: page 3 – 22

Assigning Attributes to Objects: page 3 – 23

Defining Attributes: page 3 – 25

Defining Foreign Keys: page 3 – 27

Defining Foreign Keys

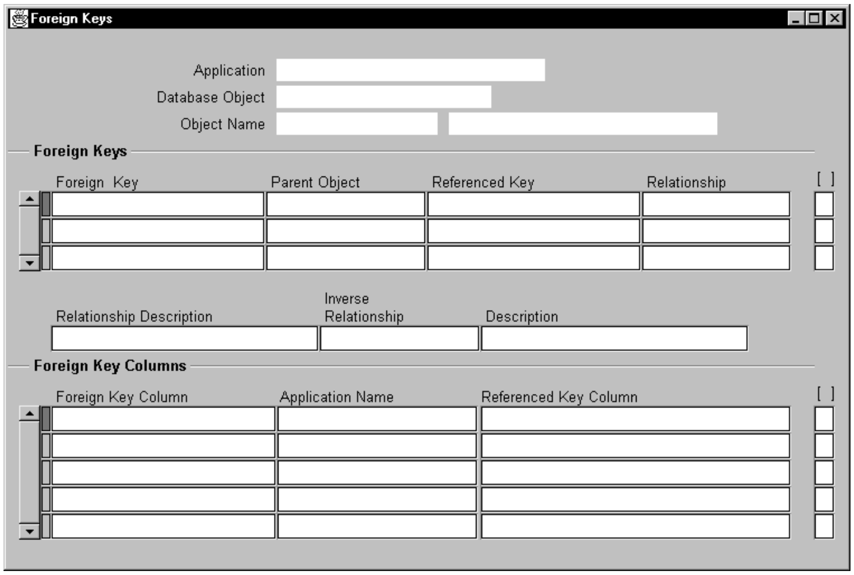
The combination of primary keys and foreign key relationships determine the navigation through your web flow. That is, if your flow must have navigation from one region to another based upon the same object, a foreign key must be defined for that object.

Prerequisites

- ❑ Define objects.
- ❑ Define attributes.
- ❑ Define object attributes.
- ❑ Define Primary Key(s).

▶ To define a foreign key:

1. In the Web Application Dictionary, navigate to the Foreign Keys window by choosing the Foreign Keys button from the Objects folder window.



2. Enter the foreign key.
3. Select the parent object (database view).
4. Enter the referenced key. This is the unique (or primary) key of the parent object.
5. Optionally, enter a description for the relationship.
6. Optionally, enter the inverse relationship.

7. Optionally, enter a description for the inverse relationship.
8. Enter the foreign key column.
9. Enter the referenced key column.
10. Repeat the last two steps until all referenced key columns have been assigned.

See Also

Defining Objects: page 3 – 22

Assigning Attributes to Objects: page 3 – 23

Defining Attributes: page 3 – 25

Defining Primary Keys: page 3 – 26

Defining Regions

Regions can be defined and then assigned to one or more pages.

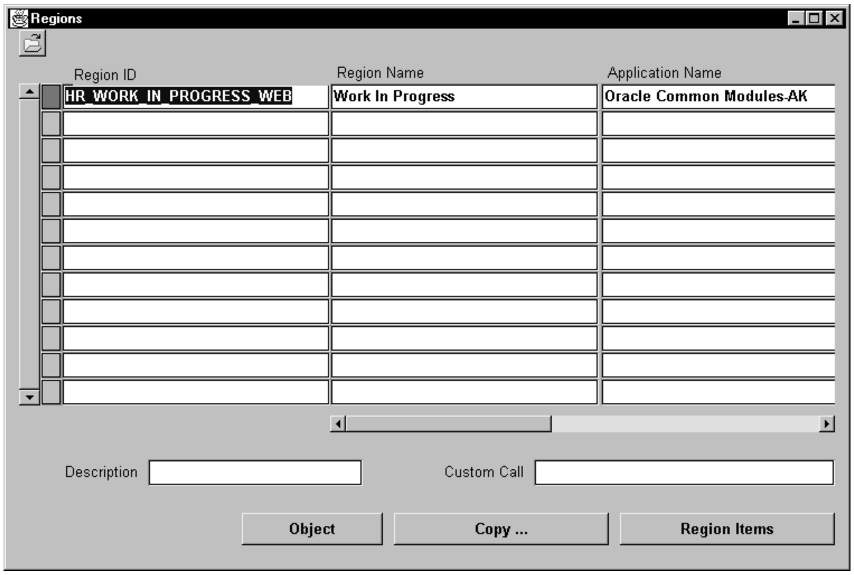
You can define regions that do not display. Such regions serve as a way of navigating from one object to another.

Prerequisites

❑ Defined objects.

▶ **To define a region:**

1. Navigate to the Regions folder window.



Region ID	Region Name	Application Name
HR_WORK_IN_PROGRESS_WEB	Work In Progress	Oracle Common Modules-AK

Description

Custom Call

2. If you want to copy an existing region to then modify and save as a new region, choose the Copy button. Enter a new application name and new region name.
3. If you are creating a new region from scratch, enter the identifier for the region.
4. Enter a user–friendly region name.
5. Enter the application associated with the region.
6. Select the object name associated with the region.
7. Select a region style: Single–row, Multi–row.
8. If you selected Single–row in the previous step, enter the number of columns (a field and its label) to display in the region before the line wraps.
9. Optionally, enter a free–form description for the region.
10. Use the Region Items button to navigate to the Region Items window.

See Also

Creating Region Items: page 3 – 31

Creating Region Items

Region items are attributes or object attributes that are placed within a region. These are typically the attributes that you want to display in the region. However, there are exceptions to this.

Prerequisites

❑ Define attributes to associate with regions.

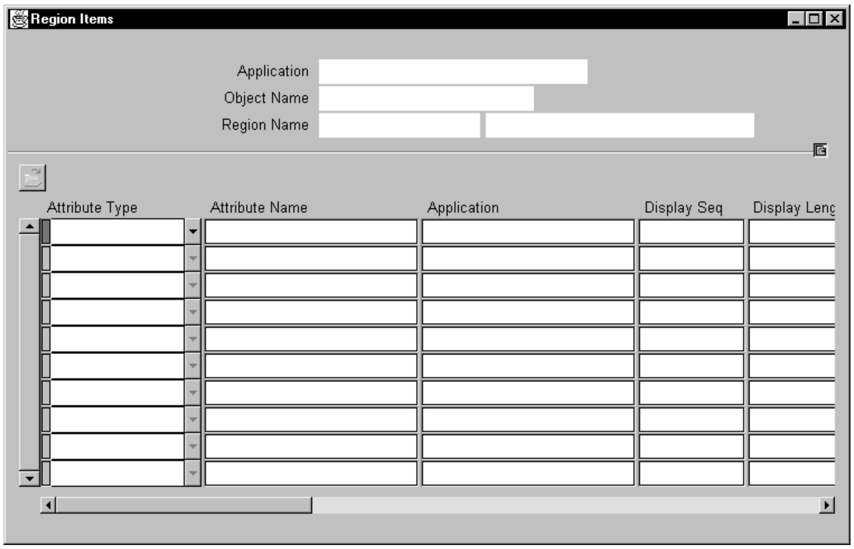
❑ Define objects.

❑ Define object attributes to associate with regions.

❑ Define regions.

▶ **To create region items:**

1. In the Web Applications Dictionary, navigate to the Region Items folder window. Do this by selecting a region in the Regions window and choosing the Region Items button.



2. Select the attribute type, either attribute or object attribute, to associate to a region. An attribute is usually reserved for use with a button.
3. Select the name of an existing attribute or object attribute.
4. Enter a display sequence for the region item. This is required.

This determines the order of the region items, whether they display or not. If you do not want a region item displayed, select the Hidden item style in the next step.

5. Select an item style, either Button or Text.

   Note: The Checkbox, Hidden, and Poplist item styles are not supported.

6. Optionally, indicate whether the region item can be queried.

7. Optionally, indicate whether the underlying column of the region item should determine the order in which data is displayed, and whether that order is ascending or descending.

   This generates a web query form.

See Also

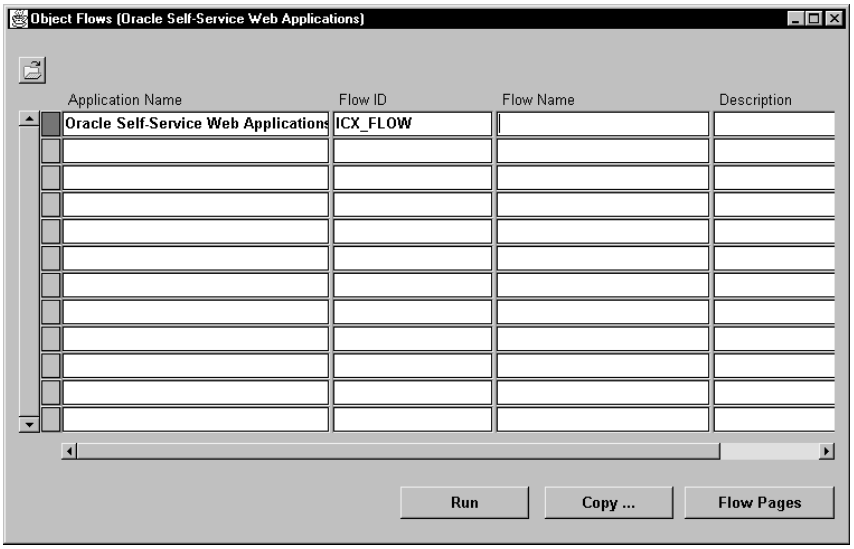
Defining Regions: page 3 – 29

Defining Object Flows

A flow is made up of web pages, each of which is made up of regions, which, in turn, is made up of region items.

▶ **To define object flows:**

1. In Web Application Dictionary, navigate to the Object Flows folder window.



Application Name	Flow ID	Flow Name	Description
Oracle Self-Service Web Applications	ICX_FLOW		

2. If you want to copy an existing flow to then modify and save as a new flow, choose the Copy button. Enter a new application name and new flow name.
3. If you are creating a new flow from scratch, enter the application name.
4. Enter a flow identifier.
5. Enter a flow name and description.

See Also

Defining Flow Pages: page 3 – 33

Defining Flow Page Regions: page 3 – 35

Defining Flow Page Relations: page 3 – 37

Defining Flow Page Region Links: 3 – 38

Defining Flow Pages

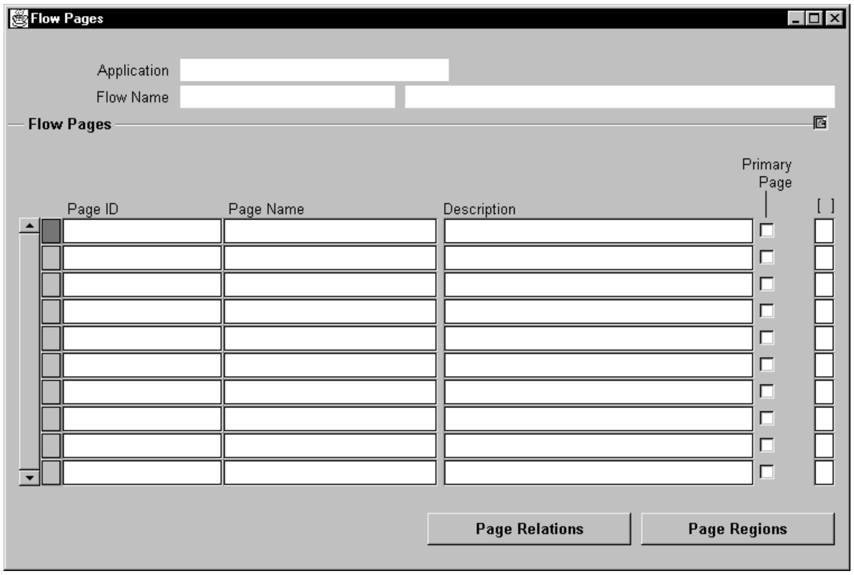
A flow page is part of a flow. Each flow may have one or more related pages. Flow pages contain one or more flow page regions.

Prerequisites

- ❑ Define objects.
- ❑ Define primary keys.
- ❑ Define foreign keys.
- ❑ Define regions.
- ❑ Define region items.
- ❑ Define object flow(s).

▶ **To define a flow page:**

1. In Web Applications Dictionary, navigate to the Flow Pages window by choosing the Flow Pages button from the Object Flows window.



2. Enter a page identifier.
3. Enter a page name and an optional description.
4. Indicate whether the page you are defining is a primary page, that is, the first page in the flow.
5. For each flow page, optionally choose the Page Relations button to define associated pages of the flow.

For each flow page, choose the Page Regions button to define the region(s) that make up the page.

See Also

Defining Object Flows: page 3 – 32

Defining Flow Page Regions: page 3 – 35

Defining Flow Page Relations: page 3 – 37

Defining Flow Page Region Links: 3 – 38

Defining Flow Page Regions

Each flow page is made up of one or more regions. A region may be defined as a primary region. Regions not defined as primary regions must have a parent region.

Prerequisites

❑ Define objects.

❑ Define primary keys.

❑ Define foreign keys.

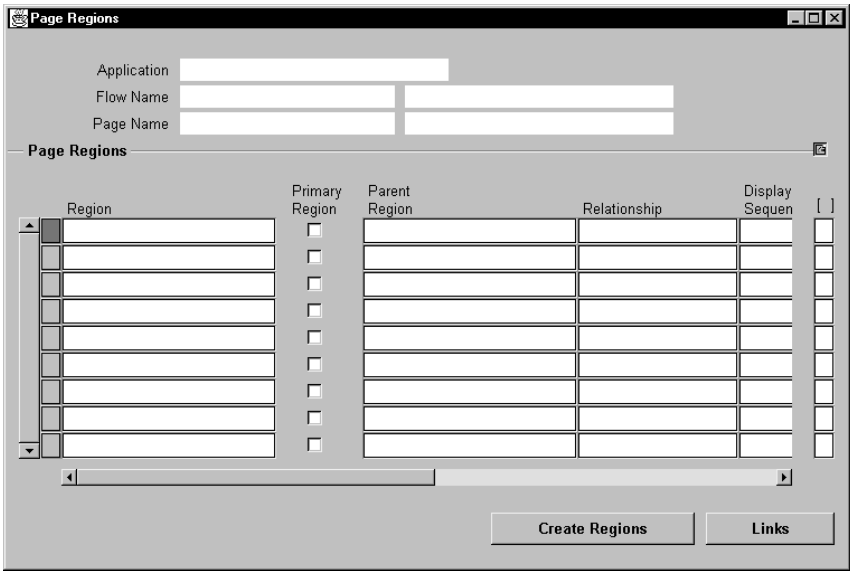
❑ Define regions.

❑ Define flow page.

❑ If you are defining multple regions on a page, a page relation must exist between the different regions. See Defining Flow Page Relations: page 3 – 37.

▶ **To define attribute navigation:**

1. In Web Application Dictionary, navigate to the Page Regions window by choosing the Page Regions button from the Flow Pages window.



2. Enter the region name. This displays in your flow at the top of the region.

   Once you select a region, the rest of the fields default from the definition of the region. You may override these defaults.

3. Indicate whether this region is a primary region.

   A primary region will have child regions, and will not have a parent region.

4. For secondary regions (those not defined as a primary region), enter the parent region. This parent region is the region that displays before the current region.

   The parent region is based upon an object that has a relationship with the secondary region object.

5. For secondary regions, enter a relationship. This was originally established when you defined the foreign keys.

6. For all regions, enter the display sequence.

See Also

Defining Regions: page 3 – 29

Defining Object Flows: page 3 – 32

Defining Flow Pages: page 3 – 33

Defining Flow Page Relations: page 3 – 37

Defining Flow Page Region Links: 3 – 38

Defining Flow Page Relations

Define the relations between pages making up a flow.

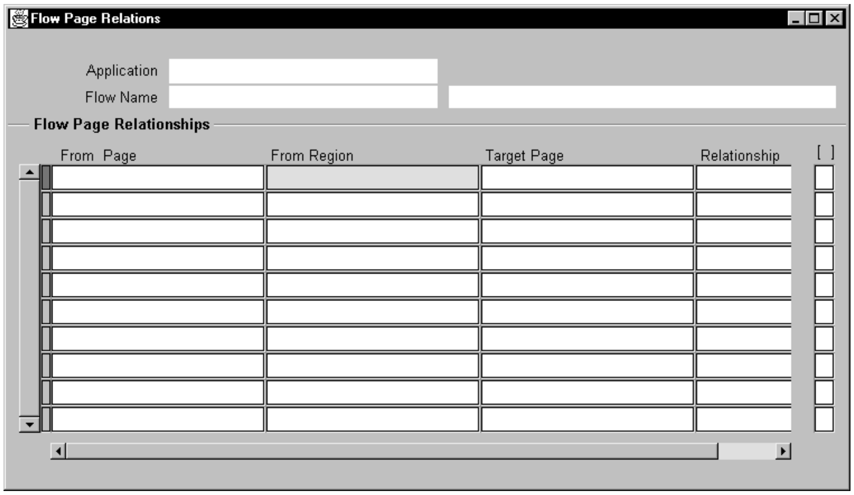
Prerequisites

❑ Define flow pages.

❑ Foreign keys must exist between the base objects (database views) used in regions.

▶ **To define a flow page relation:**

1. In Web Application Dictionary, navigate to the Page Relations window by choosing the Page Relations button from the Flow Pages window.



2. Select a from page and a from region.

   The from page and from region you select is the page and corresponding region from which a hypertext link is executed to display the current page.

3. Select the relationship.

   This was originally defined when you defined foreign keys.

See Also

Defining Regions: page 3 – 29

Defining Object Flows: page 3 – 32

Defining Flow Pages: page 3 – 33

Defining Flow Page Regions: page 3 – 35

Defining Flow Page Region Links: 3 – 38

Defining Flow Page Region Links

If applicable, you must define hypertext links for flow page regions.

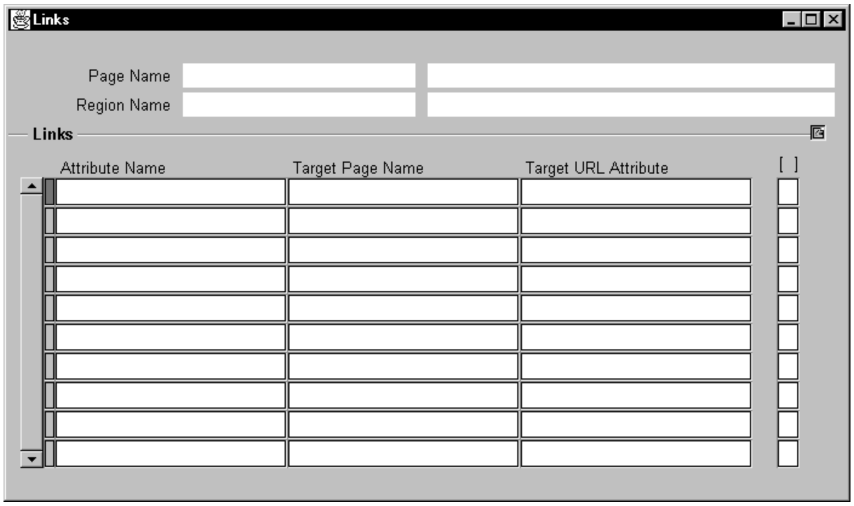
Prerequisites

❑ Define flow page regions.

❑ Define flow page relations.

▶ **To define attribute navigation:**

1. In Web Application Dictionary, navigate to the Links window by choosing the Links button from the Page Regions window.



2. Select an attribute name.
3. Enter a target page name or a target URL (Uniform Resource Locator) to link to. For external web sites, you must use a target URL.

   The target URL attribute is the column in the view that contains the URL. The URL takes the format of the following example:

   ```
   http://www.oracle.com
   ```

See Also

Defining Object Flows: page 3 – 32

Defining Flow Pages: page 3 – 33

Defining Flow Page Regions: page 3 – 35

Defining Flow Page Relations: page 3 – 37

CHAPTER

4 Predefined Inquiry Flows

This chapter describes the predefined inquiry flows installed with Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers including the following topics:

- Predefined Inquiry Flows: page 4 – 2
- Viewing and Modifying an Inquiry Flow: page 4 – 4
- Inquiries: page 4 – 5

Predefined Inquiry Flows

This section describes the predefined inquiry flows installed with Oracle Web Customers, Oracle Web Employees, and Oracle Web Suppliers.

You can use the information below to customize the predefined inquiries, or to help you design your own custom inquiries.

Many inquiries have predefined securing and / or excluding attributes. These are listed below with the description of the inquiry. There may also be securing or excluding attributes predefined at the responsibility level. For a list of predefined securing and excluding attributes at the responsibility level, see: Data Security: page 1 – 15.

Inquiry Flows by Product

The following table lists the inquiry flows and their associated products Web Customers, Web Employees, and / or Web Suppliers.

Inquiry Flow	Web Customers	Web Employees	Web Suppliers
Business Summary	✓		
Credit Memos	✓		
Customer Products Summary		✓	
Customer Projects		✓	✓
Customer Summary		✓	
Demand by (Sales) Order Number	✓		
Demand by Item	✓		
Depot Repairs	✓		
Engineering Change Order		✓	
Event Calendar	✓		
Expense Reports		✓	
Installed Base	✓		
Invoice History	✓		

Table 4 – 1 Inquiry flows and associated products (Page 1 of 3)

Inquiry Flow	**Web Customers**	**Web Employees**	**Web Suppliers**
Invoices and Debit Memos	✓		
Invoices Received			✓
Item Forecasts	✓		
Key Member Projects		✓	
List of Users		✓	
Margin Analysis by Customer		✓	
Margin Analysis by Item		✓	
Margin Analysis by Sales Representative		✓	
Margin Analysis by Territory		✓	
On–time Delivery Performance			✓
Open Delivery Schedules			✓
Open Requisition by Users		✓	
Open Requisition Lines		✓	
Overdue Receipts			✓
Payment History	✓		
Payments Sent			✓
Planner Projects		✓	
Purchase Order Change History		✓	✓
Purchase Orders			✓
Receipt History			✓
Repairs Summary	✓		
Requests for Quotes			✓
Requisition by Users		✓	

Table 4 – 1 Inquiry flows and associated products (Page 2 of 3)

Inquiry Flow	Web Customers	Web Employees	Web Suppliers
Requisition Distributions		✓	
Requisition History		✓	
Returns			✓
Sales Orders	✓		
Schedule Summary			✓
Service Requests Summary	✓		
Supplier Agreements			✓
Supplier Catalog Summary		✓	✓
Supplier Item Summary			✓
Supplier Planning Schedule			✓
Supplier Shipping Schedule			✓

Table 4 – 1 Inquiry flows and associated products (Page 3 of 3)

Viewing and Modifying an Inquiry Flow

Each of the predefined inquiries has a 1st flow page ID and a 1st region ID that you use to query the details using the Web Applications Dictionary. For example, the Margin Analysis by Customer's 1st flow page ID is ICX_CUSTOMER_MARGIN; the 1st region ID is ICX_90DAY_ANALYSIS_BY_CUSTOMER.

Each inquiry retrieves data from one or more Oracle Applications. For example, the Outbound Request for Quotes retrieves data from Oracle Purchasing.

> **Note:** You can view the HTML source to obtain the flow code, page code, and region code for that page. You can then query for details using Web Applications Dictionary.

▶ **To modify or view an inquiry flow:**

1. Log in to Oracle Applications and select the Applications for the Web Manager responsibility.
2. In the Web Applications Dictionary, open the Flows workbench.
3. Query for the flow ID, ”ICX_INQUIRIES.”
4. Select ICX_INQUIRIES and choose the Flow Pages button.
5. In the Flow Pages window, query for a flow page using the 1st flow page ID from one of the inquiries listed below.
6. Use the 1st region ID displayed to query the related regions, links, and so on.

See Also

Defining Flow Pages: page 3 – 33

Defining Flow Page Regions: page 3 – 35

Defining Flow Page Relations: page 3 – 37

Defining Flow Page Region Links: 3 – 38

Inquiries

Automotive Inquiries

The automotive inquiries provide information on demand transactions passed through the Oracle Demand Stream Processor to the internal demand manager and the customer’s supply manager. With these inquiries, you can view details of each transaction, including transaction date and time, whether the transaction was processed without errors, the number of records processed, the quantity of demand by customer item, and the individual demand records with job sequence and job number.

View Demand by Item

Provides summary information on like item orders between customers and suppliers.

1st Flow Page ID ICX DEMAND BY ITEMS

1st Region ID	ICX TRANSMISSIONS BY ITEMS
Application(s) Accessed	Oracle Automotive
Securing Attribute(s)	ICX_CUSTOMER_ORG_ID, ICX_CUSTOMER_SITE_ID. See: Data Security: page 1 – 15.

View Demand by Order Number

Provides information by order number.

1st Flow Page ID	ICX DEMAND BY SO
1st Region ID	ICX TRANSMISSIONS BY SO
Application(s) Accessed	Oracle Automotive
Securing Attribute(s)	ICX_CUSTOMER_ORG_ID, ICX_CUSTOMER_SITE_ID. See: Data Security: page 1 – 15.

Customer Summary Inquiry

Provides a view of the entire customer relationship, including profile information and order information such as sales orders, returns, cancellations, and service orders. With this inquiry, customers can view their own profile, service coverage, order status, service requests, and so on. Field service and sales personnel can look up information on their customers, and senior management can view a complete customer profile.

1st Flow Page ID	ICX_PANEL_CUSTOMERS_ALL
1st Region ID	ICX_CUSTOMER_LIST
Application(s) Accessed	Oracle Service, Order Entry, Receivables
Securing Attribute(s)	ICX_CUSTOMER_ORG_ID. See: Data Security: page 1 – 15.

Engineering Change Order (ECO) Inquiry

Employees and suppliers can view ECO information, including revised items and components. This inquiry also provides detailed information derived from Oracle Purchasing, Order Entry, Planning, and Service.

1st Flow Page ID	ENG_ECO_REVISIONS
1st Region ID	ENG_ECO_HEADER
Application(s) Accessed	Oracle Engineering, Purchasing, Order Entry, Master Scheduling / MRP and Supply Chain Planning, and Service

Event Calendar Inquiry

Provides a listing of all events for which your customers may register.

1st Flow Page ID	ICX_SM_EVENTS_1
1st Region ID	ICX_SM_EVENTS_1

Expense Reports Inquiry

Provides general and detail information on expense reports.

1ST Flow Page ID	ICX_AP_EXP_RPT_D
1st Region ID	ICX_AP_EXP_RPT_NEW_D
Application(s) Accessed	Oracle Payables

Item Forecasts Inquiry

Provides information on items in a particular forecast.

1st Flow Page ID	ICX_MRP_FORECASTS_D
1st Region ID	ICX_MRP_FORECASTS_D
Application(s) Accessed	Oracle Master Scheduling / MRP and Supply Chain Planning
Securing Attribute(s)	ICX_CUSTOMER_ORG_ID. See: Data Security: page 1 – 15.

List of Users Inquiry

Provides a list of the users that have prepared the requisition.

1st Flow Page ID	ICX_PO_LIST_USERS
1st Region ID	ICX_PO_LIST_USERS
Application(s) Accessed	Oracle Purchasing

Margin Analysis Inquiries

The Margin Analysis inquiries are extensions to the Margin Analysis report in Oracle Cost Management. The Margin Analysis inquiry describes how parts of the business contribute to the gross margin of the company. See: Margin Analysis Report, *Oracle Cost Management User's Guide, Release 11*.

Margin Analysis by Customer

This inquiry enables you to analyze margin analysis data by customer over the past 90 days, one month, and one week.

1st Flow Page ID	ICX_CUSTOMER_MARGIN
1st Region ID	ICX_90DAY_ANALYSIS_BY_CUSTOMER
Application(s) Accessed	Oracle Cost Management

Margin Analysis by Item

This inquiry enables you to analyze margin analysis data by item over the past 90 days, one month, and one week.

1st Flow Page ID	ICX_ITEM_MARGIN
1st Region ID	ICX_90DAY_ANALYSIS_BY_ITEM
Application(s) Accessed	Oracle Cost Management

Margin Analysis by Sales Representative

This inquiry enables you to analyze margin analysis data by sales representative over the past 90 days, one month, and one week.

1st Flow Page ID	ICX_SALESREP_MARGIN
1st Region ID	ICX_90DAY_ANALYSIS_BY_SALESREP
Application(s) Accessed	Oracle Cost Management

Margin Analysis by Territory

This inquiry enables you to analyze margin analysis data by territory over the past 90 days, one month, and one week.

1st Flow Page ID	ICX_TERRITORY_MARGIN

1st Region ID	ICX_90DAY_ANALYSIS_TERRITORY
Application(s) Accessed	Oracle Cost Management

Outbound Request for Quote Inquiry

Suppliers can view and respond to quote requests published by the enterprise.

1st Flow Page ID	ICX_ALL_OPEN_RFQ
1st Region ID	RFQ HEADER SUMMARY
Application(s) Accessed	Oracle Purchasing
Securing Attribute(s)	ICX_SUPPLIER_ORG_ID, ICX_SUPPLIER_SITE_ID, ICX_SUPPLIER_CONTACT_ID. See: Data Security: page 1 – 15.

Partner Information Inquiry

Business Summary

1st Flow Page ID	ICX_CUSTOMERS_SUMMARY
1st Region ID	CUSTOMER PANEL

Payments and Credits Inquiries

Credit Memos

1st Flow Page ID	ICX_AR_TRX_CM_D
1st Region ID	ICX_AR_TRX_CM_D
Application(s) Accessed	Oracle Receivables

Invoice History

1st Flow Page ID	ICX_AR_TRX_INV_D
1st Region ID	ICX_AR_TRX_INV_D

Application(s) Accessed	Oracle Receivables

Invoices and Debit Memos

1st Flow Page ID	ICX_AR_TRX_INV_DM_D
1st Region ID	ICX_AR_TRX_INV_DM_D
Application(s) Accessed	Oracle Receivables

Payment History

1st Flow Page ID	ICX_AR_CASH_RECEIPTS_D
1st Region ID	ICX_AR_CASH_RECEIPTS_D
Application(s) Accessed	Oracle Receivables

Products and Orders Inquiry

Sales Orders

1st Flow Page ID	ICX_OE_HEADERS_D
1st Region ID	ICX_OE_HEADERS_D
Application(s) Accessed	Oracle Order Entry / Shipping

Project Manufacturing Inquiries

Customers and employees can access project–related sales information, and suppliers can access project–related purchase order and requisition information.

Customer Projects

Provides customer access to project–related sales order information.

1st Flow Page ID	ICX_CUSTOMER_PROJECTS
1st Region ID	ICX_CUSTOMER_PROJ_SUM
Application(s) Accessed	Oracle Project Manufacturing (Purchasing, Order Entry, Master Scheduling / MRP and Supply Chain

	Planning, Work in Process, Inventory, Projects, Cost Management)
Securing Attribute(s)	ICX_CUSTOMER_ORG_ID. See: Data Security: page 1 – 15.

Key Member Projects

Used by an external project member to access project–related information.

1st Flow Page ID	ICX_KEY_MEMBER_PROJECTS
1st Region ID	ICX_KEY_MEMBER_PROJ_SUM
Application(s) Accessed	Oracle Project Manufacturing (Purchasing, Order Entry, Master Scheduling / MRP and Supply Chain Planning, Work in Process, Inventory, Projects, Cost Management)
Securing Attribute(s)	ICX_HR_PERSON_ID. See: Data Security: page 1 – 15.

Planner Projects

Used by a project manager at a customer site to access project information by planner.

1st Flow Page ID	ICX_PLANNER_PROJECT
1st Region ID	ICX_PLANNER_PROJ_SUM
Application(s) Accessed	Oracle Project Manufacturing (Purchasing, Order Entry, Master Scheduling / MRP and Supply Chain Planning, Work in Process, Inventory, Projects, Cost Management)
Securing Attribute(s)	ICX_HR_PERSON_ID. See: Data Security: page 1 – 15.

Purchase Order Change History Inquiry

Provides historical information on purchase order changes to employees, suppliers, and customers. Making this information available on the web can decrease calls related to purchase order changes. Employees can avoid reversing purchasing decisions and make future decisions more intelligently. Employees, suppliers, and customers can share information rapidly, reducing lead and cycle times involved in your purchasing operation.

1st Flow Page ID	ICX_REVISED_PO_ALL
1st Region ID	ICX_HEADERS_ARCHIVE_CHANGES
Application(s) Accessed	Oracle Purchasing, Payables
Excluding Attribute(s)	DISTRIBUTION_NUM
Securing Attribute(s)	ICX_SUPPLIER_ORG_ID, LEVEL_ALTERED. See: Data Security: page 1 – 15.

Requisition Inquiries

The requisition inquiries provide header, line, and historical information to purchasing buyers, planners, and managers.

Open Requisition by Users

Provides header information on open requisitions by user.

1st Flow Page ID	ICX_PO_OPEN_REQS_BY_USERS
1st Region ID	ICX_PO_OPEN_REQS_BY_USERS
Application(s) Accessed	Oracle Purchasing

Open Requisition Lines

Provides line information on open requisitions by user.

1st Flow Page ID	ICX_RQS_OPEN_1
1st Region ID	ICX_RQS_OPEN_1
Application(s) Accessed	Oracle Purchasing

Requisition by Users

Provides requisition header information by user.

1st Flow Page ID	ICX_PO_REQS_BY_USERS
1st Region ID	ICX_PO_REQS_BY_USERS
Application(s) Accessed	Oracle Purchasing

Requisition Distributions

Displays General Ledger account distribution information on requisition lines.

1st Flow Page ID	ICX_RQS_LINES_DIST
1st Region ID	ICX_RQS_LINES_DIST
Application(s) Accessed	Oracle Purchasing

Requisition History

Provides historical requisition information.

1st Flow Page ID	ICX_RQS_HISTORY_1
1st Region ID	ICX_RQS_HISTORY_1
Application(s) Accessed	Oracle Purchasing

Service and Support Inquiries

Depot Repairs

1st Flow Page ID	CS_DR_SUMMARY
1st Region ID	CS_REPAIRS_SUMMARY
Application(s) Accessed	Oracle Service

Installed Base

1st Flow Page ID	CS_CP_SUMMARY
1st Region ID	CS_CP_SUMMARY
Application(s) Accessed	Oracle Service

Service Request Inquiries

Customer Products Summary

Provides a summary of service requests by customer product.

1st Flow Page ID	CS_CP_SUMMARY
1st Region ID	CS_CP_SUMMARY
Application(s) Accessed	Oracle Service
Securing Attribute(s)	ICX_CUSTOMER_CONTACT_ID, ICX_CUSTOMER_ORG_ID. See: Data Security: page 1 – 15.

Repairs Summary

Provides a summary of repairs to a particular product.

1st Flow Page ID	CS_DR_SUMMARY
1st Region ID	CS_REPAIRS_SUMMARY
Application(s) Accessed	Oracle Service
Securing Attribute(s)	ICX_CUSTOMER_CONTACT_ID, ICX_CUSTOMER_ORG_ID. See: Data Security: page 1 – 15.

Service Requests Summary

Provides a summary of service requests on a particular product.

1st Flow Page ID	CS_INCIDENT_SUMMARY
1st Region ID	CS_INCIDENT_SUMMARY
Application(s) Accessed	Oracle Service
Securing Attribute(s)	ICX_CUSTOMER_CONTACT_ID, ICX_CUSTOMER_ORG_ID. See: Data Security: page 1 – 15.

Supplier Catalog Summary Inquiry

Displays available items and number of items ordered from the catalog.

1st Flow Page ID	ICX_CATALOG_SUMMARY
1st Region ID	ICX_CATALOG_SUMMARY_R
Application(s) Accessed	Oracle Purchasing
Securing Attribute(s)	ICX_SUPPLIER_ORG_ID. See: Data Security: page 1 – 15.

Supplier Schedule Inquiry

Suppliers can view planning and shipping schedules, including consolidated forecast and release information.

1st Flow Page ID	ICX_SHIP_PLAN_SCHEDULE
1st Region ID	ICX_CHV_HEADERS_SUM
Application(s) Accessed	Oracle Supplier Scheduling

Supplier Service Inquiries

Invoices Received

1st Flow Page ID	ICX_AP_INVOICES_D
1st Region ID	ICX_AP_INVOICES_D
Application(s) Accessed	Oracle Payables

On–time Delivery Performance

1st Flow Page ID	ICX_RCV_DLVRY_PERF_D
1st Region ID	ICX_RCV_DLVRY_PERF_D
Application(s) Accessed	Oracle Receivables

Open Delivery Schedules

1st Flow Page ID	ICX_PO_EXPECT_RCPTS_D
1st Region ID	ICX_PO_EXPECT_RCPTS_D
Application(s) Accessed	Oracle Purchasing

Overdue Receipts

1st Flow Page ID	ICX_PO_LATE_RCPTS_D
1st Region ID	ICX_PO_LATE_RCPTS_D
Application(s) Accessed	Oracle Purchasing

Payments Sent

1st Flow Page ID	ICX_AP_CHECKS_D
1st Region ID	ICX_AP_CHECKS_D
Application(s) Accessed	Oracle Payables

Purchase Orders

1st Flow Page ID	ICX_PO_HEADERS_D
1st Region ID	ICX_PO_HEADERS_D
Application(s) Accessed	Oracle Purchasing

Receipt History

1st Flow Page ID	ICX_RCTP_HISTORY_D
1st Region ID	ICX_RCTP_HISTORY_D

Returns

1st Flow Page ID	ICX_RCV_RETURNS_D
1st Region ID	ICX_RCV_RETURNS_D
Application(s) Accessed	Oracle Receivables

Supplier Agreements

1st Flow Page ID	ICX_PO_SUP_AGREE_D
1st Region ID	ICX_PO_SUP_AGREE_D
Application(s) Accessed	Oracle Purchasing

Supplier Item Summary

1st Flow Page ID	ICX_PO_SUP_ITEMS_D
1st Region ID	ICX_PO_SUP_ITEMS_D
Application(s) Accessed	Oracle Purchasing

Supplier Planning Schedule

1st Flow Page ID	ICX_SHIPPING_SCHEDULE
1st Region ID	ICX_CHV_HEADERS_SUM
Application(s) Accessed	Oracle Order Entry / Shipping

Supplier Shipping Schedule

1st Flow Page ID	ICX_SHIPPING_SCHEDULE
1st Region ID	ICX_CHV_HEADERS_SUM
Application(s) Accessed	Oracle Order Entry / Shipping

CHAPTER

5

Application Programmable Interfaces

This chapter documents Oracle Self–Service Web Applications open Application Programmable Interfaces (APIs), including the following topics:

- Application Programmable Interfaces: page 5 – 2
- API Specifications: page 5 – 2
- Standard API Parameters: page 5 – 40

Application Programmable Interfaces

The following document describes the Oracle Self–Service Web Applications application programmable interfaces (API).

API Specifications

Functions

Package	FND_FORM_FUNCTIONS_PKG
File	AFFMFUNS.pls / AFFMFUNB.pls
Functionality	This package is used to insert, update, and delete Oracle Self–Service Web Applications functions.

Procedures

INSERT_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_ROWID	Out	ROWID	Y	
X_FUNCTION_ID	In	Number	Y	Get value from fnd_form_functions_s sequence
X_FUNCTION_NAME	In	Varchar2	Y	Up to 30 character long function code, suggest using upper_case
X_APPLICATION_ID	In	Number	Y	Application ID of the function must be inserted
X_FORM_ID	In	Number	N	FormID for Release 11 function NULL for Web functions
X_PARAMETERS	In	Varchar2	N	If your function does not have parameters, pass in NULL value.
X_TYPE	In	Varchar2	Y	WWW for web functions

Table 5 – 1 (Page 1 of 2)

Parameter	IN/OUT	Datatype	Required	Notes
X_WEB_HOST_NAME	In	Varchar2	N	Web host name of up to 80 characters long. NULL for no web host name
X_WEB_AGENT_NAME	In	Varchar2	N	Web agent name of up to 80 characters long. NULL for no web agent name.
X_WEB_HTML_CALL	In	Varchar2	N	Web HTML call of up to 240 characters long. Pass in NULL for no web HTML call.
X_WEB_ENCRYPT_PARAMETERS	In	Varchar2	Y	’Y’ if the parameter is encrypted, ’N’ if not
X_WEB_SECURED	In	Varchar2	Y	’Y’ if web secured, NULL if not
X_USER_FUNCTION_NAME	In	Varchar2	Y	Up to 80 character function name
X_DESCRIPTION	In	Varchar2	Y	Up to 240 character function description
X_CREATION_DATE	In	Date	Y	*sysdate*
X_CREATED_BY	In	Number	Y	
X_LAST_UPDATE_DATE	In	Date	Y	*sysdate*
X_LAST_UPDATED_BY	In	Number	Y	
X_LAST_UPDATE_LOGIN	In	Number	Y	

Table 5 – 1 (Page 2 of 2)

UPDATE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_FUNCTION_ID	In	Number	Y	Function_id of the function to be modified
X_FUNCTION_NAME	In	Varchar2	Y	Function_code up to 30 characters long
X_APPLICATION_ID	In	Number	Y	Application ID of function to be modified
X_FORM_ID	In	Number	N	FormID for Release 11 function NULL for Web functions
X_PARAMETERS	In	Varchar2	N	New parameter value for the function, NULL if none
X_TYPE	In	Varchar2	Y	WWW for web functions
X_WEB_HOST_NAME	In	Varchar2	N	Web host name of up to 80 characters long. NULL for no web host name
X_WEB_AGENT_NAME	In	Varchar2	N	Up to 80 characters long web agent name, NULL if none
X_WEB_HTML_CALL	In	Varchar2	N	Up to 240 characters long web HTML call, NULL if none
X_WEB_ENCRYPT_PARAMETERS	In	Varchar2	Y	'Y' if parameters encrypted or 'N' if not
X_WEB_SECURED	In	Varchar2	Y	'Y' if web secured or NULL if not

Table 5 – 2 (Page 1 of 2)

Parameter	IN/OUT	Datatype	Required	Notes
X_USER_FUNCTION_NAME	In	Varchar2	Y	User function name of up to 80 characters
X_DESCRIPTION	In	Varchar2	Y	New description of up to 240 characters, NULL if none
X_LAST_UPDATE_DATE	In	Date	Y	*sysdate*
X_LAST_UPDATED_BY	In	Number	Y	
X_LAST_UPDATE_LOGIN	In	Number	Y	

Table 5 – 2 (Page 2 of 2)

DELETE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_FUNCTION_ID	In	Number	Y	Function_ID of the function to be deleted

Table 5 – 3 (Page 1 of 1)

Menus

Package	FND_MENUS_PKG
File	AFMNMNUS.pls / AFMNMNUB.pls
Functionality	This package is used to insert, update, and delete Oracle Self–Service Web Applications menus.
	FND_MENUS, FND_MENUS_TL

INSERT_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_ROWID	Out	ROWID	Y	
X_MENU_ID	In	Number	Y	Menu ID for menu to be inserted
X_MENU_NAME	In	Varchar2	Y	menu_code up to 30 characters long
X_USER_MENU_NAME	In	Varchar2	Y	Displayed name of menu up to 80 characters
X_DESCRIPTION	In	Varchar2	Y	Up to 240 characters menu description
X_CREATION_DATE	In	Date	Y	*sysdate*
X_CREATED_BY	In	Number	Y	
X_LAST_UPDATE_DATE	In	Date	Y	*sysdate*
X_LAST_UPDATED_BY	In	Number	Y	
X_LAST_UPDATE_LOGIN	In	Number	Y	

Table 5 – 4 (Page 1 of 1)

UPDATE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_MENU_ID	In	Number	Y	Menu ID for menu to be updated
X_MENU_NAME	In	Varchar2	Y	menu_code up to 30 characters long
X_USER_MENU_NAME	In	Varchar2	Y	Displayed name of menu up to 80 characters
X_DESCRIPTION	In	Varchar2	Y	Up to 240 character responsibility description

Table 5 – 5 (Page 1 of 2)

Parameter	IN/OUT	Datatype	Required	Notes
X_CREATION_DATE	In	Date	Y	*sysdate*
X_CREATED_BY	In	Number	Y	
X_LAST_UPDATE_DATE	In	Date	Y	*sysdate*
X_LAST_UPDATED_BY	In	Number	Y	
X_LAST_UPDATE_LOGIN	In	Number	Y	

Table 5 – 5 (Page 2 of 2)

DELETE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_MENU_ID	In	Number	Y	Menu ID of the menu to be deleted

Table 5 – 6 (Page 1 of 1)

Menu Entries

Package	FND_MENU_ENTRIES_PKG
File	AFMNENTS.pls / AFMNENTB.pls
Functionality	This package is used to insert, update, and delete Oracle Self–Service Web Applications menu entries.
Tables	FND_MENU_ENTRIES, FND_MENU_ENTRIES_TL

INSERT_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_ROWID	Out	ROWID	Y	
X_MENU_ID	In	Number	Y	Menu ID for menu to be inserted
X_ENTRY_SEQUENCE	In	Number	Y	Display sequence of entry in menu
X_SUB_MENU_ID	In	Number	Y	Menu ID if entry is a menu
X_FUNCTION_ID	In	Number	Y	Function ID if entry is a form function
X_PROMPT	In	Varchar2	Y	Displayed name of entry, 30 characters
X_DESCRIPTION	In	Varchar2	Y	Up to 240 characters for menu entry description
X_CREATION_DATE	In	Date	Y	*sysdate*
X_CREATED_BY	In	Number	Y	
X_LAST_UPDATE_DATE	In	Date	Y	*sysdate*
X_LAST_UPDATED_BY	In	Number	Y	
X_LAST_UPDATE_LOGIN	In	Number	Y	

Table 5 – 7 (Page 1 of 1)

UPDATE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_MENU_ID	In	Number	Y	Menu ID for menu to be updated
X_ENTRY_SEQUENCE	In	Number	Y	Display sequence of entry in menu

Table 5 – 8 (Page 1 of 2)

Parameter	IN/OUT	Datatype	Required	Notes
X_SUB_MENU_ID	In	Number	Y	Menu ID if entry is a menu
X_FUNCTION_ID	In	Number	Y	Function ID if entry is a form function
X_PROMPT	In	Varchar2	Y	Displayed name of entry, 30 characters
X_DESCRIPTION	In	Varchar2	Y	Up to 240 character menu entry description
X_CREATION_DATE	In	Date	Y	*sysdate*
X_CREATED_BY	In	Number	Y	
X_LAST_UPDATE_DATE	In	Date	Y	*sysdate*
X_LAST_UPDATED_BY	In	Number	Y	
X_LAST_UPDATE_LOGIN	In	Number	Y	

Table 5 – 8 (Page 2 of 2)

DELETE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_MENU_ID	In	Number	Y	Menu ID for menu to be deleted
X_ENTRY_SEQUENCE	In	Number	Y	Display sequence of entry to be deleted

Table 5 – 9 (Page 1 of 1)

Responsibilities

Package	FND_RESPONSIBILITY_PKG
File	AFSCRSPS.pls / AFSCRSPB.pls

Functionality This package is used to insert, update, and delete Oracle Self–Service Web Applications responsibilities.

Tables FND_RESPONSIBILITY, FND_RESPONSIBILITY_TL

Procedures

INSERT_ROW( )

Parameter	IN/OUT	Datatype	Required	Default	Notes
X_ROWID	Out	ROWID	Y		
X_APPLICATION_ID	In	Number	Y		Application ID for responsibility to be inserted
X_RESPONSIBILITY_ID	In	Number	Y		Responsibility ID for responsibility to be inserted
X_RESPONSIBILITY_KEY	In	Varchar2	Y		Responsibility_code up to 30 characters
X_WEB_AGENT_NAME	In	Varchar2	N		Web agent name of the responsibility up to 80 characters, NULL if none
X_WEB_HOST_NAME	In	Varchar2	N		Web host name of the responsibility up to 80 characters, NULL if none
X_DATA_GROUP_APPLICATION_ID	In	Number	N	NULL	
X_DATA_GROUP_ID	In	Number	N	NULL	
X_MENU_ID	In	Number	N	NULL	
X_START_DATE	In	Date	N		Start date of responsibility is not implemented for web responsibilities. NULL.

Table 5 – 10 (Page 1 of 2)

Parameter	IN/OUT	Datatype	Required	Default	Notes
X_END_DATE	In	Date	N		End date of responsibility is not implemented for web responsibilities. NULL.
X_GROUP_APPLICATION_ID	In	Number	N		Irrelevant to web responsibility. NULL.
X_REQUEST_GROUP_ID	In	Number	N		Irrelevant to web responsibility. NULL.
X_VERSION	In	Varchar2	Y		'W' for web responsibility
X_RESPONSIBILITY_NAME	In	Varchar2	Y		Up to 100 characters responsibility name
X_DESCRIPTION	In	Varchar2	Y		Up to 240 characters responsibility description
X_CREATION_DATE	In	Date	Y		*sysdate*
X_CREATED_BY	In	Number	Y		
X_LAST_UPDATE_DATE	In	Date	Y		*sysdate*
X_LAST_UPDATED_BY	In	Number	Y		
X_LAST_UPDATE_LOGIN	In	Number	Y		

Table 5 – 10 (Page 2 of 2)

UPDATE_ROW( )

Parameter	IN/OUT	Datatype	Required	Default	Notes
X_APPLICATION_ID	In	Number	Y		Application ID of the responsibility
X_RESPONSIBILITY_ID	In	Number	Y		Responsibility ID of the responsibility

Parameter	IN/ OUT	Datatype	Required	Default	Notes
X_RESPONSIBILITY_KEY	In	VarChar2	Y		Responsibility_code up to 30 characters
X_WEB_AGENT_NAME	In	Varchar2	N		Up to 80 characters long new web agent name
X_WEB_HOST_NAME	In	Varchar2	N		Up to 80 characters long new web host name
X_DATA_GROUP_APPLICATION_ID	In	Number	N	NULL	
X_DATA_GROUP_ID	In	Number	N	NULL	
X_MENU_ID	In	Number	N	NULL	
X_START_DATE	In	Date	N		Start date of responsibility is not implemented for web responsibilities. NULL.
X_END_DATE	In	Date	N		End date of responsibility is not implemented for web responsibilities. NULL.
X_GROUP_APPLICATION_ID	In	Number	N		Irrelevant to web responsibilities. NULL.
X_REQUEST_GROUP_ID	In	Number	N		Irrelevant to web responsibilities. NULL.
X_VERSION	In	Varchar2	Y		Suggested value ’W’ for web responsibility
X_RESPONSIBILITY_NAME	In	Varchar2	Y		Up to 100 characters long new responsibility name
X_DESCRIPTION	In	Varchar2	Y		Up to 240 characters long description, NULL if none
X_LAST_UPDATE_DATE	In	Date	Y		*sysdate*

Table 5 – 11 (Page 2 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
X_LAST_UPDATED_BY	In	Number	Y		
X_LAST_UPDATE_LOGIN	In	Number	Y		

Table 5 – 11 (Page 3 of 3)

DELETE_ROW( )

Parameter	IN/ OUT	Datatype	Required	Notes
X_APPLICATION_ID	In	Number	Y	Application ID of the responsibility to be deleted
X_RESPONSIBILITY_ID	In	Number	Y	Responsibility ID of the responsibility to be deleted

Table 5 – 12 (Page 1 of 1)

Responsibility – Securing Attributes Association

Package	ICX_RESP_SEC_ATTR_PVT
File	ICXVTRSS.pls / ICXVTRSB.pls
Functionality	This package is used to associate and dissassociate securing attributes with responsibilities.
Tables	AK_RESP_SECURITY_ATTRIBUTES

☞ **Attention:** This is a standard Oracle API package. For a detailed description of the standard parameters, refer to the Standard API Parameters section below.

Create_Resp_Sec_Attr( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API .G_ FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize.
p_simulate	In	Varchar2	N	FND_API .G_ FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API .G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API .G_VALI D_LEVEL _FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages

Table 5 – 13 (Page 1 of 2)

Parameter	**IN/ OUT**	**Datatype**	**Required**	**Default**	**Notes**
p_responsibility_id	In	Number	Y		Responsibility ID for the responsibility to which the securing attribute will to be associated
p_application_id	In	Number	Y		Application ID for the responsibility to which the securing attribute will be associated
p_attribute_code	In	Varchar2	Y		Attribute code for the securing attribute that will be associated to the responsibility
p_attribute_appl_id	In	Number	Y		Attribute application ID for the securing attribute that is to be associated to the responsibility
p_created_by	In	Number	Y		
p_creation_date	In	Date	Y		*sysdate*
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	Y		

Table 5 – 13 (Page 2 of 2)

Delete_Resp_Sec_Attr( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages

Table 5 – 14 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_responsibility_id	In	Number	Y		Responsibility ID of the responsibility for which the securing attribute must be disassociated
p_application_id	In	Number	Y		Application ID of the responsibility for which the securing attribute must be disassociated
p_attribute_code	In	Varchar2	Y		Attribute code of the securing attribute which must be disassociated from the responsibility
p_attribute_appl_id	In	Number	Y		Attribute application ID of the securing attribute which must be disassociated from the responsibility

Table 5 – 14 (Page 2 of 2)

Responsibility – Excluding Attributes Association

Package	ICX_RESP_EXCL_ATTR_PVT
File	ICXVTRES.pls / ICXVTREB.pls
Functionality	This package is used to associate and disassociate excluding attributes with responsibilities.
Tables	AK_EXCLUDED_ITEMS

☞ **Attention:** This is a standard Oracle Applications API package. For a detailed description of the standard parameters, refer to the ”Standard API Parameters” section below.

Create_Resp_Excl_Attr( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages

Table 5 – 15 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_responsibility_id	In	Number	Y		Responsibility ID of the responsibility to which the excluding attribute must be added
p_application_id	In	Number	Y		Application ID of the responsibility to which the excluding attribute must be added
p_attribute_code	In	Varchar2	Y		Attribute code of the excluding attribute which must be added to the responsibility
p_attribute_appl_id	In	Number	Y		Attribute application ID of the excluding attribute which must be added to the responsibility
p_created_by	In	Number	Y		
p_creation_date	In	Date	Y		*sysdate*
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	Y		

Table 5 – 15 (Page 2 of 2)

Delete_Resp_Excl_Attr( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages

Table 5 – 16 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_responsibility_id	In	Number	Y		Responsibility ID of the responsibility for which the excluding attribute must be disassociated
p_application_id	In	Number	Y		Application ID of the responsibility for which the excluding attribute must be disassociated
p_attribute_code	In	Varchar2	Y		Attribute code of the excluding attribute which must be disassociated from the responsibility
p_attribute_appl_id	In	Number	Y		Attribute application ID of the excluding attribute which must be disassociated from the responsibility

Table 5 – 16 (Page 2 of 2)

Web Users

Package	FND_USER_PVT
File	AFSVWUSS.pls / AFSVWUSB.pls
Functionality	Used to insert, update, and delete web users.
Tables	FND_USER

☞ **Attention:** This is a standard Oracle Applications API package. For a detailed description of the standard parameters, refer to the ”Standard API Parameters” section below.

Create_User( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_customer_contact_id	In	Number	N	NULL	Customer contact ID for the web user to be created

Table 5 – 17 (Page 1 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_date_format_mask	In	Varchar2	N	DD–MON –YYYY	Date format mask of the web user
p_email_address	In	Varchar2	N	NULL	Email address of the web user up to 240 characters
p_end_date_active	In	Date	N	NULL	The expiration date of this user account, NULL if no expiration date
p_internal_contact_id	In	Number	N	NULL	Internal contact ID for the web user to be created
p_known_as	In	Varchar2	N	NULL	Username as appeared on the screen, up to 80 characters
p_language	In	Varchar2	N	AMERICAN	User's default language after login in a multi–language environment
p_last_login_date	In	Date	N	NULL	Last login date of this user account
p_limit_connects	In	Number	N	NULL	Upper limit (number) for the times a user is allowed to access the database in a single session
p_limit_time	In	Number	N	NULL	Upper limit (in hours) a user can be logged in before a session expires
p_password	In	Varchar2	Y		User password of up to 80 characters long
p_supplier_contact_id	In	Number	N	NULL	Supplier contact ID for the user to be registered

Table 5 – 17 (Page 2 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_host_port	In	Varchar2	Y		*host:port* of web listener connected to database. Necessary to access AOL security code from PL/SQL.
p_username	In	Varchar2	Y		User name of up to 80 characters
p_created_by	In	Number	Y		
p_creation_date	In	Date	Y		*sysdate*
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	N	NULL	
p_user_id	Out	Number	Y		User ID as generated by the API on behalf of the calling procedure from FND_USER_S sequence

Table 5 – 17 (Page 3 of 3)

Delete_User( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller

Table 5 – 18 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_user_id	In	Number	Y		The user ID of the user to be deleted

Table 5 – 18 (Page 2 of 2)

Update_User( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize

Table 5 – 19 (Page 1 of 4)

Parameter	**IN/ OUT**	**Datatype**	**Required**	**Default**	**Notes**
p_simulate	In	Varchar2	N	FND_API.G_FALSE	Pass in ’T’ if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_FALSE	Pass in ’T’ if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_user_id	In	Number	Y		The user ID of the user to be updated
p_customer_contact_id	In	Number	N	FND_API.G_MISS_NUM	New customer contact ID for the web user, if default value, API will not change the old contact ID
p_date_format_mask	In	Varchar2	N	FND_API.G_MISS_CHAR	New date format mask of the web user, if default value, API will not change the old date format mask

Table 5 – 19 (Page 2 of 4)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_email_address	In	Varchar2	N	FND_API.G_MISS_CHAR	New email address, if default value, API will not change the old email address
p_end_date_active	In	Date	N	FND_API.G_MISS_DATE	New expiration date of this user account, if default value, API will not change the old expiration date
p_internal_contact_id	In	Number	N	FND_API.G_MISS_NUM	New internal contact ID of the user, if default value, API will not change the old internal contact ID of the user
p_known_as	In	Varchar2	N	FND_API.G_MISS_CHAR	New known_ as of up to 80 characters, if default value, API will not change the old known as value of the user
p_language	In	Varchar2	N	FND_API.G_MISS_CHAR	New default language code for the user, if default value, API will not change the old language code
p_last_login_date	In	Date	N	FND_API.G_MISS_DATE	New last login date, if default value, API will not change the old login date
p_limit_connects	In	Number	N	FND_API.G_MISS_NUM	New upper limit of database connections, if default value, API will not change the old limit
p_limit_time	In	Number	N	FND_API.G_MISS_NUM	New upper limit in hours of the connection time, if default value, API will not change the old limit

Table 5 – 19 (Page 3 of 4)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_host_port	In	Varchar2	Y	FND_API.G_MISS_NUM	*host:port* of web listener connected to database. This is needed to access AOL security code from PL/SQL.
p_old_password	In	Varchar2	N	FND_API.G_MISS_CHAR	Old password. If default value, API will not change the old password.
p_new_password	In	Varchar2	N	FND_API.G_MISS_CHAR	New password. If default value, API will not change the old password.
p_supplier_contact_id	In	Number	N	FND_API.G_MISS_NUM	New supplier contact ID, if default value, API will not change old supplier contact ID
p_username	In	Varchar2	N	FND_API.G_MISS_CHAR	New username, if default value, API will not change old user name. Oracle recommends you not change the username because of the password encryption mechanism.
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	N	NULL	

Table 5 – 19 (Page 4 of 4)

User Profile

Package	ICX_USER_PROFILE_PVT
File	ICXVUPFS.pls / ICXVUPFB.pls
Functionality	Used to insert, update, and delete Oracle Self–Service Web Applications user profiles.

Tables ICX_USER_PROFILES

☞ **Attention:** This is a standard Oracle Applications API package. For a detailed description of the standard parameters, refer to the ”Standard API Parameters” section below.

Procedures

Create_Profile( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_ FALSE	Pass in ’T’ if you want to initialize the error message list. Pass in ’F’ if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_ FALSE	Pass in ’T’ if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in ’T’ if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list

Table 5 – 20 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_user_id	In	Number	Y		ID of user for which profile is created
p_days_needed_by	In	Number	N	NULL	Number of days of the preferred delivery
p_req_default_template	In	Varchar2	N	NULL	Default template of requisition up to 25 characters long
p_req_override_loc_flag	In	Varchar2	N	NULL	’Y’ or ’N’
p_req_override_req_code	In	Varchar2	N	NULL	Override requisition code up to 25 characters long
p_created_by	In	Number	Y		
p_creation_date	In	Date	Y		*sysdate*
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	Y		

Table 5 – 20 (Page 2 of 2)

Update_Profile( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_AP I.G_ FALSE	Pass in ’T’ if you want to initialize the error message list. Pass in ’F’ if you do not want to initialize

Table 5 – 21 (Page 1 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_simulate	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_user_id	In	Number	Y		User ID for which user the profile is to be updated
p_days_needed_by	In	Number	N	FND_API.G_MISS_NUM	New preferred days of delivery. If default value, API will not change the old days needed by value.
p_req_default_template	In	Varchar2	N	FND_API.G_MISS_CHAR	New default template. If default value, API will not change the old requisition default template.

Table 5 – 21 (Page 2 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_req_override_loc_flag	In	Varchar2	N	FND_API.G_MISS_CHAR	New override location flag, 'Y' or 'N'. If default value, API will not update the override location flag.
p_req_override_req_code	In	Varchar2	N	FND_API.G_MISS_CHAR	New override requisition code, up to 25 characters. If default value, API will not update the override requisition code.
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	Y		

Table 5 – 21 (Page 3 of 3)

Delete_Profile( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller

Table 5 – 22 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_user_id	In	Number	Y		The user ID of the user profile to be deleted

Table 5 – 22 (Page 2 of 2)

User – Responsibility Association

Package	FND_USER_RESPONSIBILITY_PKG
File	AFSCURSS.pls / AFSCURSB.pls
Functionality	Used for associating responsibilities with web users.
Tables	FND_USER_RESPONSIBILITY

INSERT_ROW( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
X_ROWID	Out	Varchar2	Y		
X_USER_ID	In	Number	Y		User ID of the user to which responsibility is to be associated
X_APPLICATION_ID	In	Number	Y		Application_ID for the responsibility that will be associated to the user
X_RESPONSIBILITY_ID	In	Number	Y		Responsibility_id for the responsibility that will be associated to the user
X_RESPONSIBILITY_KEY	In	VarChar2	Y		Responsibility_code up to 30 characters
X_START_DATE	In	Date	N		Not implemented, suggested value: NULL
X_END_DATE	In	Date	N		Not implemented, suggested value: NULL
X_DESCRIPTION	In	Varchar2	Y		Description of user–responsibility association of up to 240 characters NULL if no description
X_WINDOW_WIDTH	In	Number	N	NULL	
X_WINDOW_HEIGHT	In	Number	N	NULL	
X_WINDOW_XPOS	In	Number	N	NULL	
X_WINDOW_YPOS	In	Number	N	NULL	
X_WINDOW_STATE	In	Varchar2	N	NULL	
X_NEW_WINDOW_FLAG	In	Varchar2	N	NULL	
X_FUNCTION1	In	Varchar2	N	NULL	
X_FUNCTION2	In	Varchar2	N	NULL	
X_FUNCTION3	In	Varchar2	N	NULL	
X_FUNCTION4	In	Varchar2	N	NULL	

Table 5 – 23 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
X_FUNCTION5	In	Varchar2	N	NULL	
X_FUNCTION6	In	Varchar2	N	NULL	
X_FUNCTION7	In	Varchar2	N	NULL	
X_FUNCTION8	In	Varchar2	N	NULL	
X_FUNCTION9	In	Varchar2	N	NULL	
X_FUNCTION10	In	Varchar2	N	NULL	
X_MODE	In	Varchar2	N	'R'	

Table 5 – 23 (Page 2 of 2)

DELETE_ROW( )

Parameter	IN/OUT	Datatype	Required	Notes
X_USER_ID	In	Number	Y	User ID of the user–responsibility association to be deleted
X_APPLICATION_ID	In	Number	Y	Application ID of the responsibility in the user–responsibility association to be deleted
X_RESPONSIBILITY_ID	In	Number	Y	Responsibility ID of the responsibility in the user–responsibility association to be deleted
X_RESPONSIBILITY_KEY	In	VarChar2	Y	Responsibility_code up to 30 characters

Table 5 – 24 (Page 1 of 1)

User – Securing Attribute Values Association

Package	ICX_USER_SEC_ATTR_PVT
File	ICXVTUSS.pls / ICXVTUSB.pls

Functionality Used for associating and disassociating securing attribute values with web users.

Tables AK_WEB_USER_SEC_ATTR_VALUES

☞ **Attention:** This is a standard Oracle Applications API package. For a detailed description of the standard parameters, refer to the "Standard API Parameters" section below.

Procedures

Create_User_Sec_Attr( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_ FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure

Table 5 – 25 (Page 1 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_web_user_id	In	Number	Y		The user ID of the user profile to be deleted
p_attribute_code	In	Varchar2	Y		Attribute code of the securing attribute to be associated to the web user
p_attribute_appl_id	In	Number	Y		Attribute application ID of the securing attribute to be associated to the web user
p_varchar2_value	In	Varchar2	Y		Up to 240 characters. VarChar2 value if the securing attribute is of VarChar2 type; NULL if the securing attribute is of other types.
p_date_value	In	Date	Y		A date value if the securing attribute is of date type; NULL if the securing attribute is of other types
p_number_value	In	Number	Y		A numeric value if the securing attribute is of Number type; NULL if the securing attribute is of other types
p_created_by	In	Number	Y		
p_creation_date	In	Date	Y		*sysdate*

Table 5 – 25 (Page 2 of 3)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_last_updated_by	In	Number	Y		
p_last_update_date	In	Date	Y		*sysdate*
p_last_update_login	In	Number	Y		

Table 5 – 25 (Page 3 of 3)

Delete_User_Sec_Attr( )

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_api_version_number	In	Number	Y		Suggested value: 1.0
p_init_msg_list	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want to initialize the error message list. Pass in 'F' if you do not want to initialize
p_simulate	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to roll back when returning to the caller
p_commit	In	Varchar2	N	FND_API.G_FALSE	Pass in 'T' if you want the database operations to be committed on returning to the caller
p_validation_level	In	Number	N	FND_API.G_VALID_LEVEL_FULL	There are no validation levels implemented for this API. The parameter is just here to conform to the standard. Therefore, it is not required.
p_return_status	Out	Varchar2	Y		Used to indicate the return status of the procedure

Table 5 – 26 (Page 1 of 2)

Parameter	IN/ OUT	Datatype	Required	Default	Notes
p_msg_count	Out	Number	Y		The error message count holds the number of error messages in the API message list
p_msg_data	Out	Varchar2	Y		Contains the error messages
p_web_user_id	In	Number	Y		Web user ID of the user for which the user–securing attribute association is to be deleted
p_attribute_code	In	Varchar2	Y		Attribute code of the attribute which the user–securing attribute association is to be deleted
p_attribute_appl_id	In	Number	Y		Application ID of the attribute which the user–securing attribute association is to be deleted
p_varchar2_value	In	Varchar2	Y		If the securing attribute is type VarChar2, the VarChar2 value; NULL if the attribute is not type VarChar2
p_date_value	In	Date	Y		If the securing attribute is type Date, the date value; NULL if the attribute is not type Date
p_number_value	In	Number	Y		If the securing attribute is type Number, the numeric value; NULL if the attribute is not type Number

Table 5 – 26 (Page 2 of 2)

Standard API Parameters

Some of the packages described here meet Oracle Applications API standards:

- ICX_RESP_SEC_ATTR_PVT
- ICX_RESP_EXCL_ATTR_PVT
- FND_WEBUSER_PVT
- ICX_USER_PROFILE_PVT
- ICX_USER_SEC_ATTR_PVT

The procedures in these packages have the following standard IN parameters:

- P_API_VERSION_NUMBER
- P_INIT_MFG_LIST
- P_SIMULATE
- P_COMMIT
- P_VALIDATION_LEVEL

and the following standard OUT parameters:

- P_RETURN_STATUS
- P_MSG_COUNT
- P_MSG_DATA

p_api_version_number

Every API must have a required IN parameter named:

```
p_api_version_number IN NUMBER;
```

The **p_api_version_number** has no default, thus all API callers must pass it in their calls.

This parameter is used by the API to compare the version numbers of incoming calls to its current version number, and return an unexpected error if they are incompatible.

p_init_msg_list

```
p_init_msg_list  IN     VARCHAR2 := FND_API.G_FALSE
```

The **p_init_msg_list** parameter allows API callers to request that the API does the initialization of the message list on their behalf, thus reducing the number of calls required by a caller to execute an API.

API callers have another choice, they can make a call to the message list utility function FND_MSG_PUB.Initialize to initialize the message list. Either way, it is the responsibility of the API caller to initialize the API message list.

The **p_init_msg_list** parameters defaults to FND_API.G_FALSE, which means that APIs will not initialize the message list unless asked by their callers.

p_simulate

```
p_simulate       IN VARCHAR2 := FND_API.G_FALSE;
```

The **p_simulate** parameter provides API callers with a ”What if” capability. If set to True, the API executes normally returning all its normal results and output parameters, but it rolls back any database changes it has performed.

The advantage of having such a parameter is to facilitate the testing of programs calling Oracle APIs. By setting this parameter to True, huge batch uploads can be tested over and over again without the need to create new test data or refresh the database.

The default for the **p_simulate** parameter is FALSE, which means perform the requested function and do not simulate.

To implement this functionality APIs checks the **p_simulate** parameter at the end of their execution and if set to TRUE, rollback to the standard start of API savepoint.

p_commit

```
p_commit         IN     VARCHAR2 := FND_API.G_FALSE;
```

In general APIs must never commit their work unless instructed by their callers. The **p_commit** parameter is used by API callers to ask the API to commit on their behalf after performing its function.

Before returning to its caller, an API checks the value of the **p_commit** parameter. If it is set to TRUE it commits its work.

An exception to the above scheme is an API that operates on multiple instances of a business object. In this case, the API commits its work every time it is done processing an instance of the business object.

The **p_simulate** parameter takes precedence over the p_commit parameter, i.e., if **p_simulate** is set to TRUE the value of **p_commit** is ignored, else the value of **p_commit** is honored.

The following code segment should be standard in all APIs:

```
IF FND_API.To_Boolean( p_simulate ) THEN
  ROLLBACK TO APIname_APItype;
ELSIF FND_API.To_Boolean( p_commit ) THEN
  COMMIT WORK;
END IF;
```

p_validation_level

```
p_validation_level     IN NUMBER :=
FND_API.G_VALID_LEVEL_FULL;
```

APIs use the **p_validation_level** parameter to determine which validation steps should be executed and which steps should be skipped. The main reason for using validation levels is to allow different application programs to use the same API and avoid duplicating some of the validation steps performed by itself.

The following predefined validation levels exist in the package FND_API in the file FNDPAPIS.PLS:

```
G_VALID_LEVEL_NONE  CONSTANT     NUMBER := 0;
G_VALID_LEVEL_FULL  CONSTANT     NUMBER := 100;
```

Notice that default for the **p_validation_level** parameter is to G_VALID_LEVEL_FULL, and it should be specified in the specification of the API

p_return_status

Every API must have an OUT scalar parameter that reports the API overall return status defined as follows:

```
p_return_status  OUT    VARCHAR2;
```

The return status of an API informs the caller about the result of the operation (or operations) performed by the API.

Variables holding return status values should be of type VARCHAR2(1).

The different possible values for an API return status are listed below:

```
Success: G_RET_STS_SUCCESS    CONSTANT VARCHAR2(1):='S';
```

A success return status means that the API was able to perform all the operations requested by its caller. A success return status may or may not be accompanied by messages in the API message list.

There is nothing special about an API performing its function successfully. Depending on the function performed by the API it may or may not add a success message to the API message list.

```
Error: G_RET_STS_ERROR  CONSTANT VARCHAR2(1):='E';
```

An error return status means that the API failed to perform some or all of the operations requested by its caller. An error return status is usually accompanied by messages describing the error (or errors) and how to fix it.

Usually, end users can take corrective actions to fix regular expected errors, such as missing attributes or invalid date ranges.

In general, most business object APIs operate on a single instance of a business object. Upon encountering an unexpected error, the API must perform the following:

- Rollback all its work.
- Add a message to the API message list describing the error it encountered.
- Stop processing, and return with a status of unexpected error.

It is worth noting that some APIs may decide to continue with some limited processing after encountering an error. For example, an API that encounters an error while validating a business object attribute may decide to continue validating the rest of the attribute before returning error to its calling program.

Some APIs perform more than one independent operation on a business object. Because those operations are independent, those APIs do not have to abort processing if one of the operations fail.

This means that an API can end up with a mix of errors and successes. In such case, the API overall return status should be Error. If it is required to report on the individual operations, then use separate OUT flags. The API should also maintain the database consistency through use of savepoints and rollbacks to be able to isolate and rollback the work done by the failing operation from the work done by the successful operation.

```
Unexpected Error: G_RET_STS_UNEXP_ERROR  CONSTANT
VARCHAR2(1):='U';
```

An unexpected error status means that the API has encountered an error condition it did not expect or could not handle. In this case, the API is unable to continue with its regular processing. Examples of such error are irrecoverable data inconsistency errors, memory errors, programming errors (like attempting a division by zero), and so on.

Usually, end users cannot correct unexpected errors. It is usually system administrators or application developers who can fix these errors.

In general, most business object APIs operate on a single instance of a business object. Upon encountering an unexpected error, the API must perform the following:

- Rollback all its work.
- Add a message to the API message list describing the error it encountered.
- Stop processing, and return with a status of unexpected error.

These values are constants defined in the package FND_API in the file FNDPAPIS.PLS.

p_msg_count and p_msg_data

The message count holds the number of messages in the API message list. If this number is one, then message data, entity and entity index should hold the message information. Of course, only APIs that use message entity and message entity index should define them as OUT parameters.

An example for calling a standard API:

```
in parameters: l_customer_contact_id, date_format_mask,
c_EMAIL_ADDRESS, l_end_date_active, l_internal_contact_id,
c_KNOWN_AS, c_NLS_LANGUAGE, c_LIMIT_CONNECTS, c_LIMIT_TIME,
c_PASSWORD1, l_supplier_contact_id, c_USERNAME
web_user_date_format varchar2(100);
return_status   varchar2(1);
msg_count       number;
msg_data        varchar2(2000);
sess_web_user   number(15);
webuser_id      number;
begin
sess_web_user := icx_sec.getID(icx_sec.PV_WEB_USER_ID);
   web_user_date_format :=
   icx_sec.getID(icx_sec.PV_DATE_FORMAT);
FND_WebUser_PVT.Create_User(p_api_version_number => 1.0,
   p_init_msg_list => ’T’,
   p_commit => ’T’,
   p_return_status => return_status,
   p_msg_count => msg_count,
   p_msg_data => msg_data,
   p_customer_contact_id => l_customer_contact_id,
   p_date_format_mask => date_format_mask,
   p_email_address => rtrim(ltrim(c_EMAIL_ADDRESS)),
   p_end_date_active =>
     to_date(l_end_date_active,web_user_date_format),
   p_internal_contact_id => l_internal_contact_id,
   p_known_as => rtrim(ltrim(c_KNOWN_AS)),
   p_language => c_NLS_LANGUAGE,
   p_last_login_date => sysdate,
   p_limit_connects => rtrim(ltrim(c_LIMIT_CONNECTS)),
   p_limit_time => rtrim(ltrim(c_LIMIT_TIME)),
   p_password => c_PASSWORD1,
   p_supplier_contact_id => l_supplier_contact_id,
   p_username => rtrim(ltrim(c_USERNAME)),
   p_created_by => sess_web_user,
   p_creation_date => sysdate,
   p_last_updated_by => sess_web_user,
   p_last_update_date => sysdate,
   p_last_update_login => sess_web_user,
   p_webuser_id => webuser_id);
```

After calling the API, if **return_status** is ’S’, the user is successfully added. If **return_status** is ’E’ or ’U’, the user is not added because of database or other errors.

An example for calling a non–standard API:

```
in parameters: C_APPLICATION_ID, responsibility_id,
   agent_name, host_name, trim_C_RESP_NAME,C_DESCRIPTION,
   version

sess_web_user varchar2(30);
row_id varchar2(30);
err_num number;
c_message varchar2(2000);
err_mesg varchar2(240);

begin
   sess_web_user := icx_sec.getID(icx_sec.PV_WEB_USER_ID);
     fnd_responsibility_pkg.insert_row(row_id,
          C_APPLICATION_ID,
          responsibility_id,
          agent_name,
          host_name,
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          '',
          sysdate,
          '',
          '',
          '',
          version,
          trim_C_RESP_NAME,
          rtrim(ltrim(C_DESCRIPTION)),
          sysdate,
          sess_web_user,
          sysdate,
          sess_web_user,
          sess_web_user);

exception
when others then

   err_num := SQLCODE;
   c_message := SQLERRM;
   select substr(c_message,12,512) into err_mesg from dual;

   icx_util.add_error(err_mesg);
   icx_admin_sig.error_screen(err_mesg);
```

Unlike standard APIs, the non–standard APIs do not have a **return_status** to indicate whether database operation was performed successfully. Therefore, it is typical to use the non–standard APIs (*xxx*_pkg) with a standard exception handler.

Index

Reader's Comment Form

Oracle Self–Service Web Applications Implementation Manual
A58294–01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information we use 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 about this manual? What did you like least about it?

If you find any errors or have any other suggestions for improvement, please indicate the topic, chapter, and page number below:

Please send your comments to:

Oracle Applications Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065 USA
Phone: (415) 506–7000 Fax: (415) 506–7200

If you would like a reply, please give your name, address, and telephone number below:

Thank you for helping us improve our documentation.