Oracle® Projects

APIs, Client Extensions, and Open Interfaces Reference

Release 11*i*

Part No. B12427–01

June 2004

ORACLE®

Oracle Projects APIs, Client Extentions, and Open Interfaces Reference Release 11*i*

Part No. B12427–01

Author: Janet Buchbinder

Major Contributing Authors: Guriqpal S. Gill, Stephen A. Gordon, Halina Matyla, Matthew Ness, Juli Anne Tolley

Contributors: Sakthivel Balasubramanian, Sandeep Bharathan, Peter Budelov, B.P. Chandrasekaran, Neeraj Garg, Srikanth Goteti, Ramesh Krishnamurthy, Jeanne Lowell, Manish Malhotra, Nikhil Mishra, Cedric Ng, Johnson Paulraj, Fiona Purves, Subramanian Venkataraman

Contents

Preface

Welcome to Release 11*i* of the *Oracle Projects APIs, Client Extensions, and Open Interfaces Reference.*

This guide contains the information you need to implement, maintain, and use the APIs, client extensions, and open interfaces that are available when you use Oracle Projects.

- Part I: Overview
  - Chapter 1, Overview of Oracle Projects APIs, Client Extensions, and Open Interfaces, gives a general description of application programming interfaces (APIs), client extensions, and open interfaces, and their use in Oracle Projects.
- Part II: Oracle Projects APIs
  - Chapters 2 through 6 give descriptions of the application programming interfaces (APIs) that are available in Oracle Projects. Chapter 2 is an overview of Oracle Projects APIs. The subsequent chapters in this section include descriptions of APIs, and instructions for using them:
  - Chapter 3: Oracle Project Foundation APIs
  - Chapter 4: Oracle Project Costing APIs
  - Chapter 5 Oracle Project Billing APIs
  - Chapter 6: Oracle Project Management APIs.
- Part III: Oracle Projects Client Extensions

- Chapter 7 is an overview of Oracle Projects client extensions. The subsequent chapters in this section include descriptions of the client extensions and instructions for using them:
- Chapter 8: Oracle Project Foundation client extensions
- Chapter 9: Oracle Project Costing client extensions
- Chapter 10 Oracle Project Billing client extensions
- Chapter 11: Oracle Project Resource Management client extensions
- Chapter 12: Oracle Project Management client extensions.

- Part IV: Oracle Projects Open Interfaces
  - Chapter 13, Oracle Projects Open Interfaces, describes the Transaction Import Open Interface.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market–leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

http://www.oracle.com/accessibility/

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither

evaluates nor makes any representations regarding the accessibility of these Web sites.

Other Information Sources

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

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

Online Documentation

All Oracle Applications documentation is available online (HTML or PDF).

- **Online Help** – Online help patches (HTML) are available on Oracle*MetaLink*.
- **About Documents** – Refer to the About Document for the mini–pack or family pack that you have installed to learn about new documentation or documentation patches that you can download. About Documents are available on Oracle*MetaLink*.

Guides Related to All Products

Oracle Applications User's Guide

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

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

Oracle Projects Documentation Set

Oracle Projects Implementation Guide

Use this manual as a guide for implementing Oracle Projects. This manual also includes appendixes covering function security, menus and responsibilities, and profile options.

Oracle Projects Fundamentals

Oracle Project Fundamentals provides the common foundation shared across the Oracle Projects products (Project Costing, Project Billing, Project Resource Management, Project Management, and Project Collaboration). Use this guide to learn fundamental information about the Oracle Projects solution.

This guide includes a Navigation Paths appendix. Use this appendix to find out how to access each window in the Oracle Projects solution.

Oracle Project Costing User Guide

Use this guide to learn detailed information about Oracle Project Costing. Oracle Project Costing provides the tools for processing project expenditures, including calculating their cost to each project and determining the GL accounts to which the costs are posted.

Oracle Project Billing User Guide

Use this guide to learn how to use Oracle Project Billing to process client invoicing and measure the profitability of your contract projects.

Oracle Project Management User Guide

This guide shows you how to use Oracle Project Management to manage projects through their lifecycles – from planning, through execution, to completion.

Oracle Project Resource Management User Guide

This guide provides you with information on how to use Oracle Project Resource Management. It includes information about staffing, scheduling, and reporting on project resources.

User Guides Related to This Product

Oracle Assets User Guide

In Oracle Assets, you can post capital project costs to become depreciable fixed assets. Refer to this guide to learn how to query mass additions imported from Oracle Projects to Oracle Assets and to review asset information.

Oracle General Ledger User Guide

Use this manual when you plan and define your chart of accounts, accounting period types and accounting calendar, functional currency, and set of books. The manual also describes how to define journal entry sources and categories so you can create journal entries for your general ledger. If you use multiple currencies, use this manual when you define additional rate types, and enter daily rates. This manual also includes complete information on implementing Budgetary Control.

Oracle HRMS Documentation Set

This set of guides explains how to define your employees, so you can give them operating unit and job assignments. It also explains how to set up an organization (operating unit). Even if you do not install Oracle HRMS, you can set up employees and organizations using Oracle HRMS windows. Specifically, the following manuals will help you set up employees and operating units:

- **Using Oracle HRMS – The Fundamentals**

  This user guide explains how to set up and use enterprise modeling, organization management, and cost analysis.

- **Managing People Using Oracle HRMS**

  Use this guide to find out about entering employees.

Oracle Inventory User Guide

If you install Oracle Inventory, refer to this manual to learn how to define project–related inventory transaction types and how to enter transactions in Oracle Inventory. This manual also describes how to transfer transactions from Oracle Inventory to Oracle General Ledger.

Oracle Payables User Guide

Refer to this manual to learn how to use Invoice Import to create invoices in Oracle Payables from Oracle Projects expense reports data in the Oracle Payables interface tables. This manual also explains how to define suppliers, and how to specify supplier and employee numbering schemes for invoices created using Oracle Projects.

Oracle Project Manufacturing Implementation Manual

Oracle Project Manufacturing allows your company to associate manufacturing costs and inventory to a specific project and task. Use this manual as your first source of information if you are implementing Oracle Project Manufacturing.

Oracle Purchasing User Guide

If you install Oracle Purchasing, refer to this user guide to read about entering and managing the requisitions and purchase orders that relate to your projects. This manual also explains how to create purchase orders from project–related requisitions in the AutoCreate Documents window.

Oracle Receivables User Guide

Use this manual to learn more about Oracle Receivables invoice processing and invoice formatting, defining customers, importing transactions using AutoInvoice, and Defining Automatic Accounting in Oracle Receivables.

Oracle Business Intelligence System Implementation Guide

This guide provides information about implementing Oracle Business Intelligence (BIS) in your environment.

BIS 11*i* User Guide Online Help

This guide is provided as online help only from the BIS application and includes information about intelligence reports, Discoverer workbooks, and the Performance Management Framework.

Using Oracle Time Management

This guide provides information about capturing work patterns such as shift hours so that this information can be used by other applications such as General Ledger.

Installation and System Administration

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before installing Oracle Applications.

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications and the technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user's guides and implementation guides.

Upgrading Oracle Applications

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

Maintaining Oracle Applications

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

Oracle Applications System Administrator's Guide

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

Oracle Alert User's Guide

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

Oracle Applications Developer's Guide

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

Other Implementation Documentation

Oracle Applications Product Update Notes

Use this guide as a reference for upgrading an installation of Oracle Applications. It provides a history of the changes to individual Oracle Applications products between Release 11.0 and Release 11*i*. It includes new features, enhancements, and changes made to database objects, profile options, and seed data for this interval.

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 you implement Oracle Projects. This manual details additional steps and setup considerations for implementing Oracle Projects with Multiple Reporting Currencies.

Multiple Organizations in Oracle Applications

This guide describes how to set up and use Oracle Projects with Oracle Applications' Multiple Organization support feature, so you can define

and support different organization structures when running a single installation of Oracle Projects.

Oracle Workflow Administrator's Guide

This guide explains how to complete the setup steps necessary for any Oracle Applications product that includes workflow–enabled processes, as well as how to monitor the progress of runtime workflow processes.

Oracle Workflow Developer's Guide

This guide explains how to define new workflow business processes and customize existing Oracle Applications–embedded workflow processes. It also describes how to define and customize business events and event subscriptions.

Oracle Workflow User's Guide

This guide describes how Oracle Applications users can view and respond to workflow notifications and monitor the progress of their workflow processes.

Oracle Workflow API Reference

This guide describes the APIs provided for developers and administrators to access Oracle Workflow.

Oracle Applications Flexfields Guide

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

Oracle eTechnical Reference Manuals

Each eTechnical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications and integrate Oracle Applications data with non–Oracle applications, and write custom reports for Oracle Applications products. Oracle eTRM is available on Oracle*MetaLink*.

Oracle Applications User Interface Standards for Forms–Based Products

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

Oracle Manufacturing APIs and Open Interfaces Manual

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

Oracle Order Management Suite APIs and Open Interfaces Manual

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

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*i*.

Training and Support

Training

Oracle offers a complete set of training courses to help you and your staff master Oracle Projects and reach full productivity quickly. These courses are organized 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 University at any of our many Education Centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization structure,

terminology, and data as examples in a customized training session delivered at your own facility.

Support

From on–site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Projects 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 Oracle server, and your hardware and software environment.

Do Not Use Database Tools to Modify Oracle Applications Data

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

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

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

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

About Oracle

Oracle provides 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 software modules for financial management, supply chain management, manufacturing, project systems, human resources management and customer relationship management.

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

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

Your Feedback

Thank you for using Oracle Projects and this implementation guide.

Oracle values your comments and feedback. At the end of this guide is a Reader's Comment Form you can use to explain what you like or dislike about Oracle Projects or this implementation guide. Mail your comments to the following address or contact your Support representative.

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

PART I: OVERVIEW

CHAPTER

1

Overview of Oracle Projects APIs, Client Extensions, and Open Interfaces

This chapter contains an overview of the APIs, Client Extensions, and Open Interfaces that are provided with the Oracle Projects applications.

Overview of Oracle Projects APIs, Client Extensions, and Open Interfaces

Oracle Projects integration tools are powerful, flexible tools that enable you to capture data from other Oracle applications or your own applications, define necessary format conversions, and direct data to Oracle Projects.

Oracle Projects applications provide application programming interfaces (APIs), client extensions, and open interfaces that enable you to:

- Import legacy data into Oracle Applications
- Link the Oracle Projects with external applications that you build, applications on other computers, and even the applications of your suppliers and customers
- Extend the functionality of Oracle Projects to conform with your business

Oracle Projects APIs

Application programming interfaces (APIs) are procedures that perform individual functions, such as creating a project based on information in an external system. The public APIs can be employed by users of Oracle Projects to integrate Oracle Projects with external systems.

APIs are called by programs that you wriite. You cannot modify the code within the APIs.

Details about the Oracle Projects APIs are provided in Section II, Oracle Projects Application Programming Interfaces (APIs).

Oracle Projects Client Extensions

Client extensions are procedures that you can modify to extend the functionality of Oracle Projects for your business needs. Each client extension procedure performs a specific task, such as deriving raw cost amounts for labor transactions.

You can modify the code of client extensions to automate your company's business rules.

Details about the Oracle Projects client extensions are provided in Section III, Oracle Projects Client Extensions.

Oracle Projects Open Interfaces

An open interface is a public API that enables you to migrate data from an external system using an interface within the product.

Oracle Projects provides the Transaction Import open interface, which enables you to load transactions from external cost collection systems into Oracle Projects.

Details about the Transaction Import are provided in Section III, Oracle Projects Open Interfaces.

PART II: APIs

CHAPTER

2

Introduction to Oracle Projects APIs

This chapter contains an introduction to the Oracle Projects APIs. It describes security requirements, error messages, and standard API parameters.

Introduction to Oracle Projects APIs

You can use the Oracle Projects APIs to integrate an external system (for example, a project management system) with Oracle Projects.

> **Note:** Some of these APIs were previously documented as Activity Management Gateway APIs. The Activity Management Gateway is no longer licensed as a separate product, but is included with Oracle Projects. All of the APIs formerly packaged as the Activity Management Gateway are described in this manual.

This section provides you with the information you need to understand the structure and processing of the public Application Programming Interfaces (APIs) provided with Oracle Projects.

This chapter provides the following information:

- Overview of the Oracle Projects APIs. This section describes some of the ways that you can use the public APIs in Oracle Projects to integrate Oracle Projects with external management systems.
- Integrating an External System with Oracle Projects. Follow the steps in this section carefully. A properly integrated system ensures that your external system can access the Oracle Projects database and that your Oracle Applications users can obtain the privileges necessary to use the application programming interfaces (APIs) discussed in this manual.
- Security Requirements. Follow the steps in this section to ensure proper security when users access Oracle Projects data from an external system.
- Handling Error Messages. This section describes how Oracle Projects APIs create error messages, and how to display them in an external application.
- Standard API Parameters. This section describes the standard input and output parameters shared by most of the public APIs in Oracle Projects.
- Common APIs. This section provides details about APIs (GET_MESSAGES, GET_DEFAULTS, and GET_ACCUM_PERIOD_INFO) that are available for use in all Oracle Projects APIs.

Overview of the Oracle Projects APIs

The Oracle Projects Application Programming Interfaces (APIs) enable you to integrate Oracle Projects with third–party systems to build a complete management tool. You can combine the functionality of your preferred system with the features of Oracle Projects, and then safely share data and exchange information.

The APIs include more than 150 application programming interfaces that:

- Perform real–time or batch sharing of data between your system and Oracle Projects, thereby eliminating duplicate data entry
- Share business rules and workflow from one system to the other
- Share setup, project planning, resource planning, budgeting, actuals, and progress data

Detailed descriptions of the APIs are provided in the detail chapters for each Oracle Projects application.

Applications of the Oracle Projects APIs

The Oracle Projects APIs are generic tools that you can use to integrate Oracle Projects with many types of external or third–party systems, including:

- **Collaborative project planning and scheduling systems.** Integrate your enterprise business systems with team–oriented project planning and scheduling tools to provide communication links throughout your company.
- **Sales management systems.** Enter your sales order using a sales management system and call APIs to create a project in Oracle Projects based on the order information.
- **Work management systems.** Use the Oracle Projects APIs to tailor a comprehensive solution that includes your work management system. Companies in the utilities industry commonly use this type of system.
- **Customer asset management and plant maintenance systems.** Share information about work orders, tasks, assets, crew labor charges, and inventory transactions charged to a project.

- **Project manufacturing systems.** Join inventory, manufacturing, and financial applications using the APIs, as Oracle's project manufacturing solution does.

Where Information Originates

The Oracle Projects APIs make two–way communication possible between Oracle Projects and a third–party external system. For example, if a purchase order issued against a task is being processed within your enterprise, you can restrict that project's task so it can't be deleted from a desktop project management system. (For more information about restricting certain actions, see: Controlling Actions in Oracle Projects: page 2 – 29.)

The following table illustrates the types of information that originates in Oracle Projects:

Information That Originates in Oracle Projects	Comments
Project templates with Quick Entry (overridable) fields	You can override some of the template's default values when you create a project.
Resources	
Organizations	
Calendars (both GL and PA periods)	
Estimate to Complete (planned for a future release)	
Actuals: cost amounts (raw and burdened), commitments (raw and burdened), quantities, revenue, PA or GL period, inception–to–date, period–to–date	Oracle Projects acts as the central repository of all project actuals, maintains common business rules (such as transaction controls), and collects a wide variety of transactions. Such transactions include phone usage records, labor, depreciation, commitments, usages, and expenses. Oracle Projects also performs complex cost burdening, generates revenue, and sends summarized information to external systems.

Table 2 – 1 Oracle Projects originating information (Page 1 of 1)

The following table illustrates the types of information that originates in an external system (in this case, a project management system).

Information That Originates in Your Project Management System	Comments
Projects and tasks of the work breakdown structure (WBS)	
Budgets: Types, Time–Phased, Amounts, Quantities, Baseline	Project managers can enter and baseline budgets from their preferred project management system or from Oracle Projects. Accounting personnel can enter budgets directly into Oracle Projects. Both types of employees can draft and update their own budget versions. Budgets created using project management systems integrate with Oracle Projects' budget calculation extensions.
Schedules and schedule changes	
Task parent reassignment	You can reassign a task to a different parent task as long the reassigned task remains under the same top task.
Percent complete: project level, WBS (any level)	Once you send this information to Oracle Projects, you can use billing extensions to produce progress billings. You can view this information in Oracle Projects using the project status inquiry (PSI) client extension.
Earned value progress reporting: Budgeted Cost of Work Scheduled, Budgeted Cost of Work Performed, Actual Cost of Work Performed, Budget at Completion	You can use earned value reporting to determine cost variance, schedule variance, and variance at completion. To view this information in Oracle Projects, use the PSI client extension.

Table 2 – 2 Project management system originating information (Page 1 of 1)

Integrating Your External System with Oracle Projects

After you install and implement Oracle Projects, you can integrate your external system with Oracle Projects. Follow the steps below to ensure that your external system can access the Oracle Projects database and that your Oracle Applications users can obtain the privileges necessary to use the APIs discussed throughout this manual.

Step 1 Create a database role

Create a special database role and assign it to anyone who will use the Oracle Projects APIs. You need to perform this step only once for each database, regardless of the number of users. Users can define their own role names. Oracle Projects provides the script **pacrrole.sql** to create and assign these database roles. The script resides in the $PA_TOP / patch / 115 / sql directory on the server and creates an output file called **pacrrole.lst**. Run the script from any directory in which you have write privileges. You run the script as any user with a Create Role privilege, such as SYSTEM or SYS. The script requires the following arguments:

- API role name, such as PMXFACE
- Username for the Oracle Applications user account, such as APPS
- Password for the Oracle Applications user account, such as APPS
- Username for the Oracle Projects user account, such as PA
- Password for the Oracle Projects user account, such as PA

From a SQL*Plus session, use the following syntax to run the script:

```
start $PA_TOP/patch/115/sql/pacrrole.sql &role &un_apps
&pw_apps &un_pa &pw_pa
```

For example, to create the role PMXFACE in the APPS account, enter:

```
start $PA_TOP/patch/115/sql/pacrrole.sql PMXFACE APPS
APPS PA PA
```

The script creates the role and grants the necessary privileges on the required database objects. Check the file **pacrrole.lst** to ensure that the script completed successfully.

Step 2 Create an Oracle Applications user

All API users must first be defined as Oracle Applications users. To define Oracle Applications users and their required responsibilities, use the Oracle Applications Release 11*i* Users window. See: *Oracle Applications System Administrator's Guide*.

Step 3 Create a database user

After you have defined an Oracle Applications user with the required responsibilities, you must create a database user. The Oracle Applications username and the database username must be identical. Oracle Projects provides the script *pacruser.sql* to create database users. The script resides in the $PA_TOP / patch / 115 / sql directory on the server. The script creates an output file called *pacruser.lst*. Run the script from any directory in which you have write privileges. You should run the script as any user with a Create User privilege, such as SYSTEM or SYS. The script requires the following arguments:

- API role name, such as PMXFACE. You must use the same role name that you created in Step 1.
- System username, such as SYSTEM
- Password for the system user, such as MANAGER
- Proposed username
- Proposed password
- Username for the Oracle Applications user account, such as APPS
- Username for the Oracle Projects user account, such as PA

From a SQL*Plus session, use the following syntax to run the script:

```
start $PA_TOP/patch/115/sql/pacruser.sql &role &sys_un
&sys_pwd &uname &pwd &un_apps &un_pa
```

To create the user JCLARK with a password of WELCOME, for example, enter:

```
start $PA_TOP/patch/115/sql/pacruser.sql PMXFACE SYSTEM
MANAGER JCLARK WELCOME APPS PA
```

Check the file **pacruser.lst** to ensure that the script completed successfully.

> **Note:** Oracle Projects provides a template script, **patempus.sql**, which facilitates the processing of large amounts of data. This script generates an output file,

pagenus.sql, while creating a large number of database users from existing Oracle Applications users. You can add WHERE conditions to narrow the criteria. Run the script from any directory in which you have write privileges. You need to run this script only for users who require access to the Oracle Projects APIs.

Caution: The Oracle Applications user is different from the database user, even if they share the same username. Each Oracle Applications user and database user has a distinct password, which you must maintain individually. Changing an Oracle Application user's password does not automatically change the database user's password. Users can choose different passwords for their Oracle Applications and database usernames.

Step 4 Set up your product in Oracle Projects

Set up your external system as a source product in Oracle Projects using the Source Products window.

Note: If the database has been exported and then imported **and** you performed Steps 1, 2, and 3 before the export/import, some or all of the grants may not work properly after the import. Use the script **pacrgran.sql** (located in $PA_TOP/patch/115/sql) to restore the grants. From a SQL*Plus session, use the following syntax to run the script:

```
start $PA_TOP/patch/115/sql/pacrgran.sql &role
&un_apps &pw_apps &un_pa &pw_pa
```

For example:

```
start $PA_TOP/patch/115/sql/pacrgran.sql PMXFACE APPS
APPS PA PA
```

Security Requirements

Each interface or application that you develop using the Oracle Projects APIs must prompt users for identifying information and then set up global variables. Follow the steps below to ensure that proper security is enforced when users access Oracle Projects data from an external system.

Step 1 Authenticate the user

Your external system should prompt users for their Oracle Projects username and password and then use this login information to establish a connection to the Oracle Projects database. After three unsuccessful attempts to establish a connection, the external system should abort and display an error message.

Step 2 Choose a responsibility

Because Oracle Applications responsibilities control users' access to Oracle Projects data, Oracle Applications users must choose a specific responsibility from the list of their valid responsibilities. Oracle Projects provides this information in the view PA_USER_RESP_V.

Column descriptions for PA_USER_RESP_V are listed in Oracle eTRM, which is available on Oracle*Metalink*.

The login username entered in Step 1 controls the Oracle Applications responsibilities retrieved by this view. Once a user chooses a responsibility, the external system also stores the corresponding USER_ID and RESPONSIBILITY_ID. The RESPONSIBILITY_NAME field is for display purposes only and need not be stored.

> **Note:** Because Oracle Applications store user names in uppercase letters, you should convert login user names to uppercase letters before using them as keys. Database connection strings are case insensitive. For example, a login username entered as "scott" is stored as "SCOTT". Typical PL/SQL code to display the responsibilities reads as follows:

```
Login Name is stored in l_login_name
l_upper_login_name = UPPER(l_login_name)
Select RESPONSIBILITY_NAME, USER_ID,
RESPONSIBILITY_ID
from PA_USER_RESP_V where
USER_NAME = l_upper_login_name
```

Caution: Do not use UPPER(USER_NAME) in the WHERE clause. Expressions used in WHERE clauses disable the index and impair performance. Always convert a value to uppercase in your code and use the converted string in the WHERE clause.

Step 3 Set up global variables

Access to Oracle Projects is controlled not only by a user's responsibility, but also by the user's organization for that responsibility. To ensure that the level of access to data matches a user's organization, use the API SET_GLOBAL_INFO to set up global variables. This API is located in the public API package PA_INTERFACE_UTILS_PUB.

SET_GLOBAL_INFO is a PL/SQL procedure that sets the global variables necessary to access data in a multi–org implemented environment.

The arguments P_RESPONSIBILITY_ID and P_USER_ID must have valid values. If the arguments contain null or invalid values, SET_GLOBAL_INFO returns an error status.

Parameters for SET_GLOBAL_INFO are shown in the following table:

Parameter	Usage	Type	Required	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RESPONSIBILITY_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the chosen responsibility (refer to Step 2: page 2 – 9)
P_USER_ID	IN	NUMBER	Yes	The identification code of the corresponding user returned by the view (refer to Step 2: page 2 – 9)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		Return status. Valid values are: S (Success), E (Error), and U (Unexpected error).
P_RESP_APPL_ID	IN	NUMBER	No	Identifier of the responsibility application

Table 2 – 3 SET_GLOBAL_INFO parameters (Page 1 of 2)

Parameter	Usage	Type	Required	Description
P_ADVANCED_PROJ_SEC_FLAG	IN	VARCHAR2	No	Flag that indicates whether to use role–based security (Default = N)
P_CALLING_MODE	IN	VARCHAR2	No	Calling mode

Table 2 – 3 SET_GLOBAL_INFO parameters (Page 2 of 2)

After completing these steps, external systems call the remaining Oracle Projects APIs necessary to complete the task, such as CREATE_PROJECT, UPDATE_PROJECT, SELECT_RESOURCE_LIST, or CREATE_DRAFT_BUDGET.

Handling Error Messages

The public APIs in Oracle Projects return applicable error messages for all updates, changes, or additions to a work breakdown structure or budget.

How Error Messages Are Created

The APIs do not stop processing when an error is encountered. Processing continues until all items are validated and error messages generated. However, if any errors are encountered during one of these processes, no records are saved to the Oracle Projects database.

The error messages contain all the information necessary to identify the data element related to each error. This information includes:

For WBS data:

- project reference
- task reference

For budget data:

- project reference
- task reference
- budget type
- budget start date

Displaying Error Messages

Because Oracle Projects APIs can be used to develop both real–time and batch integrations with external systems, display of error messages must be handled in the external application.

Use the API PA_INTERFACE_UTILS_PUB.get messages to retrieve the error messages. For details on this API and an example of PL / SQL code to retrieve the error messages, see: GET_MESSAGES: page 2 – 24.

API Messages

The following table shows the messages used in Oracle Projects APIs.

New Message Code	Length	Description	Token(s)
PA_ALL_WARN_NO_EMPL_REC_AMG	27	This user is not yet registered as an employee.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_ALL_WARN_NO_EMPL_REC_AMG	27	This user is not yet registered as an employee	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_AMT_ALLOC_LT_ACCR_AMG	27	Total amount allocated cannot be less than amount accrued or billed.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_BASE_RES_LIST_EXISTS_AMG	30	You cannot change the resource list for a baselined budget	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_CORE_NO_VERSION_ID_AMG	28	A budget does not exist for this project with specified budget type.	PROJECT_NUMBER
PA_BU_INVALID_NEW_PERIOD_AMG	28	You cannot copy a budget to a period which is out of the range of system defined periods (for example, PA period or GL period).	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_NO_BUDGET_AMG	20	There are no budget lines in this draft budget. The budget must be entered before baseline.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_NO_PROJ_END_DATE_AMG	26	Project does not have a start date or a completion date.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_NO_TASK_PROJ_DATE_AMG	27	Task does not have a start date or a completion date.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_UNBAL_PROJ_BUDG_AMG	25	Project funding is not equal to the budget total. To baseline a draft budget, the budget total must be as same as funding total.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_BU_UNBAL_TASK_BUDG_AMG	25	Task funding is not equal to the budget total of the task. To baseline a draft budget, the budget total must be as same as funding total.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE

Table 2 – 4 (Page 1 of 6)

New Message Code	Length	Description	Token(s)
PA_COPY_PROJECT_FAILED_AMG	26	Error occurred while creating the project.	PROJECT_NUMBER
PA_CREATE_CONTACTS_FAILED_AMG	29	Error occurred while creating Customer Contact information.	PROJECT_NUMBER
PA_CUST_NOT_OVERRIDABLE_AMG	27	You cannot override the Customer field while using this template.	PROJECT_NUMBER
PA_DESC_NOT_OVERRIDABLE_AMG	27	You cannot override the Description field while using this template.	PROJECT_NUMBER
PA_GET_CUST_INFO_FAILED_AMG	27	Error occurred while getting Customer information.	PROJECT_NUMBER
PA_HAS_REV/INV_AMG	18	Distribution rule cannot be changed since cost/revenue/invoices exist[\n] [\n] Cause:[\t]You cannot change the distribution rule because the project has costed items, revenue, or invoices.	PROJECT_NUMBER
PA_INVALID_DIST_RULE_AMG	24	Distribution Rule is invalid.	PROJECT_NUMBER
PA_INVALID_ORG_AMG	18	Organization is invalid.	PROJECT_NUMBER
PA_INVALID_PT_CLASS_ORG_AMG	27	Invalid organization. You cannot use the specified organization to create projects of this project type class. Choose a different organization or add the project type class to the current organization.	PROJECT_NUMBER
PA_NO_BILL_TO_ADDRESS_AMG	25	Active primary Bill To Address does not exist for the specified customer.	PROJECT_NUMBER
PA_NO_BILL_TO_CONTACT_AMG	25	Active primary billing contact does not exist for the specified customer.	PROJECT_NUMBER
PA_NO_CLIENT_EXISTS_AMG	23	The billing allocation across project client(s) is incomplete.	PROJECT_NUMBER
PA_NO_CONTACT_EXISTS_AMG	28	Billing contact not defined for each customer.	PROJECT_NUMBER
PA_NO_MANAGER_AMG	24	Project manager not currently defined for this project.	PROJECT_NUMBER
PA_NO_ORIG_PROJ_ID_AMG	22	Original project ID is not specified.	PROJECT_NUMBER
PA_NO_PROJ_CREATED_AMG	22	New project not created. No project information in the source project.	PROJECT_NUMBER
PA_NO_PROJ_ID_AMG	17	Project ID not specified.	PROJECT_NUMBER
PA_NO_REQ_CATEGORY_EXISTS_AMG	29	All mandatory class categories have not been classified.	PROJECT_NUMBER
PA_NO_SHIP_TO_ADDRESS_AMG	25	Active primary Ship To Address does not exist for the specified customer.	PROJECT_NUMBER
PA_NO_TASK_COPIED_AMG	21	No task is copied since there are tasks in the source project.	PROJECT_NUMBER
PA_NO_TASK_ID_D_AMG	19	You cannot delete this task since no task information has been provided.	PROJECT_NUMBER, TASK__NUMBER

Table 2 – 4 (Page 2 of 6)

New Message Code	Length	Description	Token(s)
PA_NO_TASK_ID_ST_AMG	20	You cannot create a subtask below this task since task information was not specified.	PROJECT_NUMBER, TASK__NUMBER
PA_NO_TOP_TASK_ID_ST_AMG	25	You cannot create a subtask below this task since task does not have top task ID.	PROJECT_NUMBER, TASK__NUMBER
PA_NO_UNIQUE_ID_AMG	19	Failed to generate unique project number. Action: Please contact your System Administrator to set up the Next Number field for Automatic Project Numbering in Implementation Options Window.	PROJECT_NUMBER
PA_PRODUCT_CODE_IS_MISSING_AMG	30	External product code required.	General
PA_PROJECT_NAME_IS_MISSING_AMG	30	Project name required.	PROJECT_NUMBER
PA_PROJECT_REF_IS_MISSING_AMG	29	External project reference required.	PROJECT_NUMBER
PA_PROJECT_STATUS_INVALID_AMG	29	The project status is invalid.	PROJECT_NUMBER
PA_PROJ_AP_INV_EXIST_D_AMG	26	You cannot delete this project since supplier invoices exist	PROJECT_NUMBER
PA_PROJ_BUDGET_EXIST_D_AMG	26	You cannot delete this project since budgets exist	PROJECT_NUMBER
PA_PROJ_BURDEN_SUM_DEST_D_AMG	29	The project is being used for the purpose of accumulating burden costs on project types.	PROJECT_NUMBER
PA_PROJ_CMT_TXN_EXIST_D_AMG	27	You cannot delete this project since project commitment transactions exist.	PROJECT_NUMBER
PA_PROJ_EVENT_EXIST_D_AMG	25	You cannot delete this project since events exist	PROJECT_NUMBER
PA_PROJ_EXP_ITEM_EXIST_D_AMG	28	You cannot delete this project since expenditure items exist.	PROJECT_NUMBER
PA_PROJ_FUND_EXIST_D_AMG	25	You cannot delete this project since funding exists.	PROJECT_NUMBER
PA_PROJ_INV_DIST_EXIST_D_AMG	28	You cannot delete this project since supplier invoice distribution lines exist	PROJECT_NUMBER
PA_PROJ_IN_USE_EXTERNAL_D_AMG	29	You cannot delete this project since project references exist	PROJECT_NUMBER
PA_PROJ_ORG_NOT_ACTIVE_AMG	26	This project organization is not active or is not within the current Project / Task owning organization hierarchy.	PROJECT_NUMBER
PA_PROJ_PO_DIST_EXIST_D_AMG	27	You cannot delete this project since purchase order distributions exist	PROJECT_NUMBER
PA_PR_COM_RUL_SET_EXIST_D_AMG	29	You cannot delete this project since compensation rules exist	PROJECT_NUMBER
PA_PR_CREATED_REF_EXIST_D_AMG	29	You cannot delete this project since compensation rule sets exist	PROJECT_NUMBER
PA_PR_INSUF_BILL_CONTACT_AMG	28	Billing contact not defined for each customer.	PROJECT_NUMBER
PA_PR_INSUF_CLASS_CODES_AMG	27	You must specify all mandatory class categories.	PROJECT_NUMBER

Table 2 – 4 (Page 3 of 6)

New Message Code	Length	Description	Token(s)
PA_PR_INSUF_PROJ_MGR_AMG	24	Project manager not currently defined for this project.	PROJECT_NUMBER
PA_PR_INVALID_START_DATE_AMG	28	Project start date must be earlier than all task start dates.	PROJECT_NUMBER
PA_PR_NAME_NOT_UNIQUE_AMG	27	Project name must be unique across all operating units in the Oracle Applications installation.	PROJECT_NUMBER
PA_PR_NAME_NOT_UNIQUE_A_AMG	25	Project name must be unique across all operating units in the Oracle Applications installation.	PROJECT_NUMBER
PA_PR_NO_PROJ_NAME_AMG	22	Project name not specified.	PROJECT_NUMBER
PA_PR_NO_PROJ_NUM_AMG	21	Project number ID not specified.	PROJECT_NUMBER
PA_PR_NO_UPD_SEGMENT1_EXP_AMG	29	You cannot change the project number since expenditure items exist	PROJECT_NUMBER
PA_PR_NUMERIC_NUM_REG_AMG	25	Please enter a numeric project number.	PROJECT_NUMBER
PA_PR_NUMERIC_NUM_REQ_AMG	25	Your implementation requires a numeric project number.	PROJECT_NUMBER
PA_PR_NUM_NOT_UNIQUE_AMG	26	Project number must be unique across all operating units in the Oracle Applications installation.	PROJECT_NUMBER
PA_PR_NUM_NOT_UNIQUE_A_AMG	24	Project number must be unique across all operating units in the Oracle Applications installation.	PROJECT_NUMBER
PA_PR_PO_REQ_DIST_EXIST_D_AMG	29	You cannot delete this project since purchase order requisitions exist.	PROJECT_NUMBER
PA_PR_START_DATE_NEEDED_AMG	27	The start date of the project is required if the completion date of the project is specified.	PROJECT_NUMBER
PA_PR_START_DATE_NEEDED_AMG	23	The start date of the project is required if the completion date of the project is specified.	PROJECT_NUMBER
PA_PUBLIC_SECTOR_INVALID_AMG	28	Invalid value for Public Sector flag.	PROJECT_NUMBER
PA_RE_ASSGMT_NOT_FOUND_AMG	26	Resource list assignment not found.	PROJECT_NUMBER
PA_RE_PROJ_NOT_FOUND_AMG	24	Specified project is invalid.	PROJECT_NUMBER
PA_RE_RL_INACTIVE_AMG	21	Resource list is not active.	PROJECT_NUMBER
PA_RE_RL_NOT_FOUND_AMG	22	Specified resource list is invalid.	PROJECT_NUMBER
PA_RE_USE_CODE_NOT_FOUND_AMG	28	Use code not found.	PROJECT_NUMBER
PA_SOURCE_TEMPLATE_INVALID_AMG	30	Source template ID is invalid.	PROJECT_NUMBER
PA_SOURCE_TEMP_IS_MISSING_AMG	30	Source template ID is required.	PROJECT_NUMBER
PA_SU_INVALID_DATES_AMG	23	From Date must be on or before the To Date.	PROJECT_NUMBER

Table 2 – 4 (Page 4 of 6)

New Message Code	Length	Description	Token(s)
PA_TASK_BURDEN_SUM_DEST_ST_AMG	30	The task is being used for the purpose of accumulating burden costs on project types.	PROJECT_NUMBER, TASK__NUMBER
PA_TASK_BURDEN_SUM_DEST_ST_AMG	29	The task is being used for the purpose of accumulating burden costs on project types.	PROJECT_NUMBER, TASK__NUMBER
PA_TASK_FUND_NO_PROJ_EVT_AMG	28	Task funding with project level events is not allowed.	PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE
PA_TASK_IN_USE_EXTERNAL_D_AMG	26	You cannot delete this task since task references exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_AP_INV_DIST_EXIST_D_AMG	30	You cannot delete this task since invoice distribution lines exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_AP_INV_DIST_EXIST_ST_AMG	30	You cannot create a subtask below this task since supplier invoice distribution lines exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_AP_INV_EXIST_D_AMG	25	You cannot delete this task since supplier invoices exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_AP_INV_EXIST_ST_AMG	26	You cannot create a subtask below this task since supplier invoices exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_ASSETASSIG_EXIST_ST_AMG	30	You cannot create a subtask below this task since assets have been assigned.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_BUDGET_EXIST_D_AMG	25	You cannot delete this task since budgets exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_BUDGET_EXIST_ST_AMG	26	You cannot create a subtask below this task since budgets exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_BUR_SCHOVR_EXIST_ST_AMG	30	You cannot create a subtask below this task since burden schedule overrides exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_CMT_TXN_EXIST_D_AMG	26	You cannot delete this task since commitment transactions exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_EBILL_RATE_EXIST_ST_AMG	30	You cannot create a subtask below this task since employee billing rate overrides exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_EVENT_EXIST_D_AMG	24	You cannot delete this task since events exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_EXP_ITEM_EXIST_D_AMG	27	You cannot delete this task since expenditure items exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_EXP_ITEM_EXIST_ST_AMG	28	You cannot create a subtask below this task since expenditure items exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_FUND_EXIST_D_AMG	27	You cannot delete this task since supplier invoice distribution line exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_JBILLTITLE_EXIST_ST_AMG	30	You cannot create a subtask below this task since job billing title overrides exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_JBILL_RATE_EXIST_ST_AMG	30	You cannot create a subtask below this task since job bill rate overrides exist.	PROJECT_NUMBER, TASK__NUMBER

Table 2 – 4 (Page 5 of 6)

New Message Code	Length	Description	Token(s)
PA_TSK_LAB_MULT_EXIST_ST_AMG	26	You cannot create a subtask below this task since there is labor multiplier for this task.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_L_COST_MUL_EXIST_ST_AMG	30	You cannot create a subtask below this task since labor cost multipliers exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_NL_BIL_RAT_EXIST_ST_AMG	30	You cannot create a subtask below this task since non–labor bill rate overrides exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_PO_DIST_EXIST_D_AMG	26	You cannot delete this task since supplier invoice distribution lines exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_PO_DIST_EXIST_ST_AMG	27	You cannot create a subtask below this task since supplier invoice distribution lines exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_PO_REQDIST_EXIST_ST_AMG	30	You cannot create a subtask below this task since purchase order requisitions exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_RULE_SET_EXIST_D_AMG	27	You cannot delete this task since compensation rule sets exist.	PROJECT_NUMBER, TASK__NUMBER
PA_TSK_TXN_CONT_EXIST_ST_AMG	28	You cannot create a subtask below this task since transaction controls exist.	PROJECT_NUMBER, TASK__NUMBER

Table 2 – 4 (Page 6 of 6)

Standard API Parameters

All Oracle Projects APIs have a set of standard input and output parameters that are used in most of the public procedures. The table below describes each of these standard API parameters.

Parameter	Usage	Type	Required	Description
P_COMMIT	IN	VARCHAR2(1)	Yes	Set this parameter to T (True) if you want the APIs themselves to issue the commit to the database. Default = F (False)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	Yes	Set this parameter to T (True) if you want to initialize the global message table. Default = F (False)
P_API_VERSION_NUMBER	IN	NUMBER	Yes	For the current version of the APIs, this parameter must be set to 1.0. This may change in future versions of the APIs.
P_RETURN_STATUS	OUT	VARCHAR2(1)		The return status of the APIs. Valid values are: S (the API completed successfully), E (business rule violation error), and U (Unexpected error, such as an Oracle error)
P_MSG_COUNT	OUT	NUMBER		Holds the number of messages in the global message table. Calling programs should use this as the basis to fetch all the stored messages. If the value for this parameter = 1, then the message code is available in P_MSG_DATA. If the value of this parameter > 1, you must use the GET_MESSAGES API to retrieve the messages.
P_MSG_DATA	OUT	VARCHAR2(2000)		Holds the message code, if the API returned only one error / warning message. Otherwise, the column is left blank.

Table 2 – 5 Standard API parameters (Page 1 of 1)

APIs That Use Composite Datatypes

Read this section if you use PL / SQL 2.3 or higher to call Oracle Projects APIs that use composite datatypes, such as an array of records

If you assign a value to a subset of variables in a PL / SQL array, first assign the values to a PL / SQL record and then add the record to the PL / SQL array. It is important to perform the steps in this order due to the way PL / SQL handles assignments to an array.

The following sample PL/SQL code shows how to assign values to the P_BUDGET_LINES_IN PL/SQL table in the CREATE_DRAFT_BUDGET API.

```
DECLARE
--variables needed for API standard parameters
l_api_version_number      NUMBER  :=1.0;
l_commit                                  VARCHAR2(1)   := 'F';
l_return_status                   VARCHAR2(1);
l_init_msg_list                   VARCHAR2(1);
l_msg_count                       NUMBER;
l_msg_data                                VARCHAR2(2000);
l_data                                            VARCHAR2(2000);
l_msg_entity                              VARCHAR2(100);
l_msg_entity_index        NUMBER;
l_msg_index                       NUMBER;
l_msg_index_out                   NUMBER;
l_encoded                                 VARCHAR2(1);
--
--variables needed for Oracle Project specific parameters
l_pm_product_code                 VARCHAR2(10);
l_pa_project_id                   NUMBER;
l_pm_project_reference    VARCHAR2(25);
l_budget_type_code        VARCHAR2(30);
l_change_reason_code              VARCHAR2(30);
l_description                             VARCHAR2(255);
l_entry_method_code               VARCHAR2(30);
l_resource_list_name              VARCHAR2(60);
l_resource_list_id        NUMBER;
l_budget_lines_in
  PA_BUDGET_PUB.budget_line_in_tbl_type;
l_budget_lines_in_rec
  PA_BUDGET_PUB.budget_line_in_rec_type;
l_budget_lines_out
  PA_BUDGET_PUB.budget_line_out_tbl_type;
l_line_index                              NUMBER;
l_line_return_status              VARCHAR2(1);
--
API_ERROR                                 EXCEPTION;
--
BEGIN
--PRODUCT RELATED DATA
l_pm_product_code :='SOMETHING';
--
--BUDGET DATA
l_pm_project_reference := 'TEST';
l_budget_type_code := 'AC';
l_change_reason_code := 'ESTIMATING ERROR';
l_description := 'New description -> 2';
```

```
l_entry_method_code := 'PA_LOWEST_TASK_BY_PA_PERIOD';
l_resource_list_id := 1001;
```

The previous example shows how to assign values to a subset of the PL/SQL table. To assign values only to PA_TASK_ID and RESOURCE_LIST_MEMBER_ID in the P_BUDGET_LINES_IN table, first assign these values to BUDGET_LINES_IN_REC and then add BUDGET_LINES_IN_REC to the BUDGET_LINES_IN PL/SQL table, as illustrated in the following example.

```
--BUDGET LINES DATA
a := 5;
FOR i IN 1..a LOOP
    if i = 1 THEN
            l_budget_lines_in_rec.pa_task_id :=1496;
            l_budget_lines_in_rec.resource_list_member_id:=1731;
    elsif i = 2 THEN
            l_budget_lines_in_rec.resource_list_member_id:=1732;
            l_budget_lines_in_rec.pa_task_id := 1495;
    elsif i = 3 THEN
            l_budget_lines_in_rec.resource_list_member_id:=1733;
            l_budget_lines_in_rec.pa_task_id := 1494;
    elsif i = 4 THEN
            l_budget_lines_in_rec.resource_list_member_id:=1734;
            l_budget_lines_in_rec.pa_task_id := 1492;
    elsif i = 5 THEN
            l_budget_lines_in_rec.resource_list_member_id:=1735;
            l_budget_lines_in_rec.pa_task_id := 1491;
end if;
            l_budget_lines_in_rec.quantity:=97;
            l_budget_lines_in_rec.period_name:=  'P06-03-95';
            l_budget_lines_in_rec.raw_cost:=300;
            l_budget_lines_in(i) := l_budget_lines_in_rec;
END LOOP;
pa_budget_pub.create_draft_budget
( p_api_version_number => l_api_version_number
 ,p_msg_count => l_msg_count
 ,p_msg_data => l_msg_data
 ,p_return_status => l_return_status
 ,p_pm_product_code => l_pm_product_code
 ,p_pa_project_id=> l_pa_project_id
 ,p_pm_project_reference => l_pm_project_reference
 ,p_budget_type_code=> l_budget_type_code
 ,p_change_reason_code => l_change_reason_code
 ,p_description => l_description
 ,p_entry_method_code => l_entry_method_code
 ,p_resource_list_name => l_resource_list_name
 ,p_resource_list_id=> l_resource_list_id
 ,p_budget_lines_in => l_budget_lines_in
 ,p_budget_lines_out => l_budget_lines_out );
```

Named Notation for Parameters

The APIs for Oracle Projects typically allow you to reference Oracle Projects entities by either identification codes or reference codes. For example, you can refer to a project using either the PROJECT_ID or the PM_PROJECT_REFERENCE.

Identification codes are usually system–generated numbers assigned to the entity by Oracle Projects. The reference code is usually a character name or description for the entity.

If a project already exists in Oracle Projects, you can reduce your processing time by passing identification codes instead of reference codes to the APIs. The APIs read identification codes and convert passed reference codes to their corresponding identification codes before execution.

If an API requires a given entity for processing, you must pass either the entity's reference code parameter or the entity's identification code parameter, but not both. If the API cannot find or derive a reference code for the required identification code parameter, the API generates an error message and aborts processing.

When passing parameters to an Oracle Projects API, you should use *named notation* (see the following example), which enables you to pass only the parameters required by a particular API. Using named notation can significantly improve the processing of update APIs.

Attention: If you pass an API parameter as NULL, the API updates the column in the database with a NULL value. If you do not want to update a column, do not pass the corresponding parameter.

Example of Named Notation

Using the API DELETE_PROJECT, you can pass either the PROJECT_ID or the PM_PROJECT_REFERENCE for the project. The following example passes the project identification code P_PA_PROJECT_ID. The SQL statement below omits optional parameters, such as P_INIT_MSG_LIST and P_COMMIT, so that they will not be updated in the table.

```
Delete_Project(p_api_version_number => 1.0
, p_msg_count => l_msg_count
, p_msg_data => l_msg_data
, p_return_status => l_return_status
, p_pm_product_code       => l_product_code
, p_pa_project_id => 1043
);
```

Data Supplied by Oracle Projects Views

The Oracle Projects APIs use identification code and reference code parameters for many Oracle Projects entities. To facilitate the retrieval of valid parameter data, selected views supply Oracle Projects data. These views are listed in the detail chapters for each Oracle Projects application.

Common APIs

The following APIs are available for use in all modules and are located in the public API package PA_INTERFACE_UTILS_PUB.

GET_MESSAGES

GET_MESSAGES is a PL/SQL procedure that retrieves messages from the message stack. If an API detects only one error during execution, the API returns the error text via the standard API output parameter P_MSG_DATA. If the API detects multiple errors, you must use the GET_MESSAGES API to retrieve the messages.

The following table shows the parameters in GET_MESSAGES.

Parameter	Usage	Type	Required	Description
P_ENCODED	IN	VARCHAR2(1)	No	Passes T (True) if you want only the message code to be returned in the p_data parameter. Default = F (False)
P_MSG_COUNT	IN	NUMBER	No	Passes the P_MSG_COUNT value returned by the API that raised the error. If P_MSG_COUNT = 1, this API returns the error text. Otherwise, this API calls the message handling package FND_MSG_PUB.
P_MSG_DATA	IN	VARCHAR2(80)	Yes	Passes the P_MSG_DATA value returned by the API that raised the error
P_DATA	OUT	VARCHAR2(2000)		The message code (if P_ENCODED = T) or the message text (if P_ENCODED = F)
P_MSG_INDEX_OUT	OUT	NUMBER		The index (cell) of the message in the global message stack
P_MSG_INDEX	IN	NUMBER	No	Message index number (default = 1)

Table 2 – 6 GET_MESSAGES parameters (Page 1 of 1)

Sample Code for Handling Multiple Messages

The following sample PL/SQL code shows how you can use GET_MESSAGES to handle multiple messages in an external application.

This example uses the procedure PA_PROJECT_PUB.create_project. You can initialize the message stack at the beginning of the session, as in this example, or for each project.

All messages are held in PL/SQL memory. For a large installation where there may be a lot of error messages, you can store all messages related to a project in a file or in the database, and initialize the message stack frequently. You use FND_MSG_PUB.initilize to initialize the message stack.

You can temporarily insert the messages into a table, as shown in the example. Or, if you are running a 'C' program or using PL/SQL file I/O utilities, you can write the messages to a log file. If you write the messages to a log file, you may want to create header information in the log file. You can then launch a text editor to instantly display the error messages.

> **Note:** The parameter *p_msg_index_out* in this code sample was added as a workaround to a known bug in Oracle AOL. This parameter may be removed in subsequent releases of Oracle Projects. If your code stops working after applying patches later than 754949, values sent as this parameter would be a likely cause.

Following is the sample code:

```
-- Initialize the message stack
FND_MSG_PUB.initialize;
pa_project_pub.create_project
    (p_api_version_number => l_api_version_number
    ,p_commit => l_commit
    ,p_init_msg_list => 'F'
    ,p_msg_count => l_msg_count
    ,p_msg_data => l_msg_data
    ,p_return_status => l_return_status
    ,p_pm_product_code => l_pm_product_code
    ,p_project_in => l_project_in_rec
    ,p_project_out => l_project_out_rec
    ,p_key_members => l_key_member_tbl
    ,p_class_categories => l_class_category_tbl
    ,p_tasks_in => l_tasks_in
    ,p_tasks_out => l_tasks_out);
IF l_return_status != 'S'
THEN
    if l_msg_count > 0 THEN
            for i in 1..l_msg_count loop
                    pa_interface_utils_pub.get_messages (
                            ,p_encoded => 'F'
                            ,p_msg_count => l_msg_count
                            ,p_msg_data => l_msg_data
```

```
                              ,p_data => l_data
                              ,p_msg_index_out => l_msg_index_out );
-- Insert the messages from l_data into error_table
Insert into error_table (error_msg) values (l_data);
           end loop;
    end if;
END IF;
```

GET_DEFAULTS

GET_DEFAULTS is a PL/SQL procedure that returns the default values required to initialize the VARCHAR2, NUMBER, and DATE variables in your programs. This API has no input parameters.

The following table shows the parameters in GET_DEFAULTS.

Parameter	Usage	Type	Description
P_DEF_CHAR	OUT	VARCHAR2(3)	Returns the default value for character variables
P_DEF_NUM	OUT	NUMBER	Returns the default value for number variables
P_DEF_DATE	OUT	DATE	Returns the default value for date variables
P_RETURN_STATUS	OUT	VARCHAR2(1)	API standard
P_MSG_COUNT	OUT	NUMBER	API standard
P_MSG_DATA	OUT	VARCHAR2(2000)	API standard

Table 2 – 7 GET_DEFAULTS parameters (Page 1 of 1)

Default values are useful when you conditionally set a value for a variable. For example, while updating a project, you may conditionally set the value for the variable L_DISTRIBUTION_RULE, depending on whether you want to update the distribution rule in Oracle Projects. To accomplish this, you would use a PL/SQL statement similar to this:

```
Pa_interface_utils.get_defaults (p_def_char => l_def_char,
                              p_def_num => l_def_num,
                              p_def_date => l_def_date,
                              p_return_status => l_return_status,
                              p_msg_count => l_msg_count,
                              p_msg_data => l_msg_data );
           l_distribution_rule := l_def_char;
```

```
l_customer_id := l_def_num;
l_end_date := l_def_date;
```

GET_ACCUM_PERIOD_INFO

GET_ACCUM_PERIOD_INFO is a PL/SQL procedure that returns information about the last period through which the project is summarized in Oracle Projects, as well as the current reporting period. Use this API to see if the actuals in your external system are current with those in Oracle Projects.

The following table shows the parameters in GET_ACCUM_PERIOD_INFO.

Parameter	Usage	Type	Required	Description
P_API_VERSION_NUMBER	IN	NUMBER	Y	API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PROJECT_ID	IN	NUMBER	Y	Unique identifier of the project
P_LAST_ACCUM_PERIOD	OUT	VARCHAR2		The period up to which the project has been summarized
P_LAST_ACCUM_START_DATE	OUT	DATE		The start date of the last summarized period
P_LAST_ACCUM_END_DATE	OUT	DATE		The end date of the last summarized period
P_CURRENT_REPORTING_PERIOD	OUT	VARCHAR2		The PA period that is defined in the current reporting period
P_CURRENT_PERIOD_START_DATE	OUT	DATE		The start date of the current reporting period
P_CURRENT_PERIOD_END_DATE	OUT	DATE		The end date of the current reporting period

Table 2 – 8 GET_ACCUM_PERIOD_INFO parameters (Page 1 of 1)

This PL/SQL example demonstrates a typical use of GET_ACCUM_PERIOD_INFO:

```
Pa_interface_utils.get_accum_period_info
    (p_api_version_number => l_api_version_number
    l_msg_count => l_msg_count,
    p_msg_data => l_msg_data,
    p_return_status => l_return_status,
    p_project_id => l_project_id,
    p_last_accum_period => l_last_accum_period,
    p_last_accum_start_date => l_last_accum_start_date,
    p_last_accum_end_date => l_last_accum_end_date,
    p_current_reporting_period => l_current_reporting_period,
    p_period_start_date => l_period_start_date,
    p_period_end_date => l_period_end_date);
```

Controlling Actions in Oracle Projects

To ensure that information in your external systems remains consistent with information in Oracle Projects, you can restrict the changes users can make to data that originates in external systems. Use the Oracle Projects Control Actions window to select the actions that you want to restrict. You can restrict these actions:

- Add Task
- Baseline Budget
- Delete Project
- Delete Task
- Update Budget
- Update Project Dates
- Update Project Description
- Update Project Name
- Update Project Number
- Update Project Organization
- Update Project Status
- Update Task Dates
- Update Task Description
- Update Task Name
- Update Task Number
- Update Task Organization

You can base the restrictions on the external system in which the information originates or on the budget type (for budget–related actions).

For example, suppose you download a project from an external system. You have a business rule that the source system always maintains project and task dates. As an additional precaution, you want to prevent users from deleting from Oracle Projects any projects and tasks that originate in an external system. To fulfill these criteria, use the Control Actions window to specify the following actions:

- Delete Project
- Delete Task
- Update Project Dates

- Update Task Dates

After you specify these actions in the Control Actions window, Oracle Projects users who try to change the project and task dates on a project that originated in an external system sees the following error message:

```
The value for this field originated in an external system. You
cannot change it.
```

A user who tries to delete the project or one of its tasks sees the following message:

```
The record originated in an external system. You cannot delete it.
```

> **Note:** You can specify effective dates for the controls you select in the Control Actions window.

Using API Procedures

The detailed chapters contain descriptions of each PL/SQL procedure used to perform certain functions in Oracle Projects based on the information you maintain in your external system.

Some APIs use composite datatypes, such as records or tables of records, as input and output parameters. Composite datatypes are PL/SQL 2.3 features that are available with Oracle 7.3.2. For more information about composite datatypes, see APIs That Use Composite Datatypes: page 2 – 19.

Tools and products that cannot use composite datatypes must call supplementary Load–Execute–Fetch APIs instead. The Load–Execute–Fetch APIs were designed without composite datatype parameters for compatibility with any tool and perform the following functions:

- Accept parameters with standard datatypes (VARCHAR2, NUMBER, and DATE) as IN parameters
- Load global composite type structures (records and tables)
- Call the underlying business object APIs (passing the global structures as IN parameters)
- Read the results from a global message and results table
- Pass the message back to the calling programs upon demand (the calling program fetches each message separately)

Call the procedures in this order:

1. **Initialize**. This step initializes the global data structures.
2. **Load.** This function loads IN parameter PL/SQL tables and records. Repeat this step until all the input structures are populated.
3. **Execute.** This step calls a business object API cover that calls the business object API. The business object API uses the global structures that were populated during the Load procedure.
4. **Fetch.** This procedure fetches one output value at a time for a business object. It also fetches messages. The calling program may or may not call the Fetch procedure, depending on the function performed.
5. **Clear.** This step clears the global structures and resets any global counters used in the calling program.

CHAPTER

3

Oracle Project Foundation APIs

This chapter describes how to implement APIs for:

- Project and task information
- Resource list and resource list member information

Project APIs

This chapter includes detailed descriptions of the APIs that you can use to integrate project data from an external system with Oracle Projects. This chapter also includes detailed descriptions of the PL/SQL procedures used to verify in real–time that:

- Project and task information you have entered into your external system is unique in Oracle Projects
- Certain functions, such as deleting a project or task, follow the business rules defined in Oracle Projects

Develop a detailed project plan using the external system you prefer. Then you can use the project APIs to push your plan into Oracle Projects and create a project based on the information in your plan. As your project plan evolves, update project information in your external system and then periodically synchronize the two systems. The project APIs update the task information and work breakdown structures (WBSs) in Oracle Projects to reflect changes made in the external system.

> **Note:** When you call any project API that requires a project identifier, you must identify the project by passing either the P_PA_PROJECT_ID or the P_PM_PROJECT_REFERENCE parameter. When you call any project API that requires a task identifier, you must identify the task by passing either the P_PA_TASK_ID or the P_PM_TASK_REFERENCE parameter.

Project API Views

The following table lists the views that provide parameter data for the project APIs. For detailed description of the views, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

View	Description
PA_CLASS_CATEGORIES_LOV_V	Retrieves class codes defined in Oracle Projects. You can use the value in the display_name field (retrieved by the PA_OVERRIDE_FIELDS_V view) to show only class codes associated with a class category. For example: ”select code description from pa_class_categories_lov_v where class_category = ’Funding Source’;”
PA_CUSTOMERS_LOV_V	Retrieves customers defined in or used by Oracle Projects

View	Description
PA_DISTRIBUTION_RULES_LOV_V	Retrieves revenue distribution rules defined in Oracle Projects
PA_KEY_MEMBERS_LOV_V	Retrieves names and employee identification numbers of team members from Oracle Projects. Note: pa_employees returns all employees defined in Oracle Projects.
PA_ORGANIZATIONS_LOV_V	Retrieves names of organizations defined in Oracle Projects
PA_OVERRIDE_FIELDS_V	Retrieves the prompts for Quick Entry fields associated with a project template. For more information about this view, see; Details about PA_OVERRIDE_FIELDS_V: page 3 – 4.
PA_OVERRIDE_FIELD_VALUES_V	Retrieves the values passed to the Quick Entry fields when a project is created
PA_PROJECT_STATUS_LOV_V	Retrieves project statuses from Oracle Projects
PA_PROJECTS_AMG_V	Retrieves project information for the organization associated with the user's responsibility
PA_SELECT_TEMPLATE_V	Retrieves project templates and projects defined in Oracle Projects
PA_SERVICE_TYPE_LOV_V	Because valid service type codes must be selected for the parameter service_type_code, you can use this view to Retrieves valid codes for service_type_code from Oracle Projects and display them in your external system.
PA_TASK_MANAGERS_LOV_V	Because valid employees must be selected for the parameter task_manager_person_id, you can use this view to retrieve valid employees from Oracle Projects and display them in your external system.
PA_TASKS_AMG_V	Retrieves information about all valid tasks for the organization associated with the user's responsibility.
PA_TASK_PROGRESS_AMG_V	Retrieves information about all valid task progress for the organization associated with the user's responsibility.
PA_STRUCT_TASKS_AMG_V	You can use this view to retrieve valid structures from Oracle Projects and display them in your external system.
PA_STRUCT_VERSIONS_LOV_AMG_V	You can use this view to retrieve valid structure versions from Oracle Projects and display them in your external system.

Table 3 – 1 Project API views (Page 2 of 2)

Details About PA_OVERRIDE_FIELDS_V

The following table shows the contents of some of the columns of the view PA_OVERRIDE_FIELDS_V, for a project with a project identification code of 1020 and all quick entry fields enabled.

ID	Field Name	Display Name	Type	Order	Req ?	View Name
1020	NAME	Project Name		20	Y	
1020	DESCRIPTION	Project Description		30	N	
1020	START_DATE	Project Start Date		40	N	
1020	COMPLETION_DATE	Project Completion Date		50	N	
1020	PROJECT_STATUS_CODE	Project Status		60	N	PA_PROJECT_STATUS_LOV_V
1020	PUBLIC_SECTOR_FLAG	Public Sector		70	N	
1020	DISTRIBUTION_RULE	Distribution Rule		80	N	PA_DISTRIBUTION_RULES_LOV_V
1020	CARRYING_OUT_ ORGANIZATION_ID	Organization		90	N	PA_ORGANIZATIONS_LOV_V
1020	KEY_MEMBER	Project Manager	PROJECT MANAGER	100	Y	PA_KEY_MEMBERS_LOV_V
1020	KEY_MEMBER	Project Coordinator	Project Coordinator	110	N	PA_KEY_MEMBERS_LOV_V
1020	CLASSIFICATION	Funding Source	Funding Source	120	Y	PA_CLASS_CATEGORIES_LOV_V
1020	CLASSIFICATION	Market Sector	Market Sector	130	N	PA_CLASS_CATEGORIES_LOV_V
1020	CUSTOMER_NAME	Customer Name	PRIMARY	140	N	PA_CUSTOMERS_LOV_V

Table 3 – 2 PA_OVERRIDE_FIELDS_V for a project with all quick entry fields enabled (Page 1 of 1)

The views you use to select valid values all have CODE and DESCRIPTION columns. Use these two columns and the value of the field LOV_VIEW_NAME to retrieve the valid values for any Quick Entry field. Valid values are stored in the CODE field. The table below shows the valid values of the quick entry fields.

Quick Entry Fields	Valid Values
NAME	
CARRYING_OUT_ ORGANIZATION_ID	PA_ORGANIZATIONS_LOV_V
PUBLIC_SECTOR_FLAG	Y or N
PROJECT_STATUS_CODE	PA_PROJECT_STATUS_LOV_V
DESCRIPTION	
START_DATE	DD–MON–YY format (e.g., 10–SEP–68)
COMPLETION_DATE	DD–MON–YY format (e.g., 13–JUL–94)
DISTRIBUTION_RULE	PA_DISTRIBUTION_RULES_LOV_V
CUSTOMER_ID	PA_CUSTOMERS_LOV_V (currently, the default CUSTOMER_ RELATIONSHIP_CODE is PRIMARY. No other value is accepted)
KEY_MEMBERS (multiple)	PA_KEY_MEMBERS_LOV_V
CLASS_CATEGORIES (multiple)	PA_CLASS_CATEGORIES_LOV_V

Table 3 – 3 Valid values for quick entry fields (Page 1 of 1)

Project API Procedures

The procedures discussed in this section are listed below. The procedures are located in the public API package PA_PROJECT_PUB.

- Project and Task Procedures
  - ADD_TASK: page 3 – 22
  - CREATE_PROJECT: page 3 – 27
  - DELETE_PROJECT: page 3 – 28
  - DELETE_TASK: page 3 – 29
  - UPDATE_PROJECT: page 3 – 31
  - UPDATE_TASK: page 3 – 36
- Load–Execute–Fetch Procedures
  - CLEAR_PROJECT: page 3 – 42
  - EXECUTE_CREATE_PROJECT: page 3 – 43

Record and Table Datatypes

The record and table datatypes used in the APIs are defined on the following pages.

PROJECT_IN_REC_TYPE Datatype

The following table shows the PROJECT_IN_REC_TYPE datatype.

Name	Type	required?	Description
PM_PROJECT_REFERENCE	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system. See Examples and Remarks: page 3 – 21.
PA_PROJECT_ID	NUMBER(15)	For update	The reference code that uniquely identifies the project in Oracle Projects
PA_PROJECT_NUMBER	VARCHAR2(25)	No	The project number that uniquely identifies the project in Oracle Projects
PROJECT_NAME	VARCHAR2(30)	Yes	Unique name of the project uniquely identifies the project in Oracle Projects
CREATED_FROM_PROJECT_ID	NUMBER(15)	Yes	Number that uniquely identifies the template from which this project originates
CARRYING_OUT_ORGANIZATION_ID	NUMBER(15)	Based on template setup	The identification code of the organization responsible for the project work
PUBLIC_SECTOR_FLAG	VARCHAR2(1)	Based on template setup	Flag that indicates whether this project is in the Public or the Private sector
PROJECT_STATUS_CODE	VARCHAR2(30)	Based on template setup	The status of the project. Any status other than CLOSED is considered active.
DESCRIPTION	VARCHAR2(250)	Based on template setup	The description of the project
START_DATE	DATE	Based on template setup	The date on which the project starts
COMPLETION_DATE	DATE	Based on template setup	The date on which the project is completed
DISTRIBUTION_RULE	VARCHAR2(30)	Based on template setup	The distribution rule that specifies the contract project's revenue accrual and billing method
CUSTOMER_ID	NUMBER(15)	Based on template setup	The identification code of the project's customer

Name	Type	required?	Description
PROJECT_RELATIONSHIP_CODE	VARCHAR2(30)	Yes	The type of customer relationship the customer has on the project
ACTUAL_START_DATE	DATE	No	The actual project start date in the external system
ACTUAL_FINISH_DATE	DATE	No	The actual project finish date in the external system
EARLY_START_DATE	DATE	No	The early project start date in the external system
EARLY_FINISH_DATE	DATE	No	The early project finish date in the external system
LATE_START_DATE	DATE	No	The late project start date in the external system
LATE_FINISH_DATE	DATE	No	The late project finish date in the external system
SCHEDULED_START_DATE	DATE	No	The scheduled project start date in the external system
SCHEDULED_FINISH_DATE	DATE	No	The scheduled project finish date in the external system
ATTRIBUTE_CATEGORY	VARCHAR2(30)	No	Used by descriptive flexfields
ATTRIBUTE1 through ATTRIBUTE10	VARCHAR2(150)	No	Descriptive flexfield
OUTPUT_TAX_CODE	VARCHAR2(30)	No	Indicates whether tax rate defined for the Project will be used for Customer Invoices.
RETENTION_TAX_CODE	VARCHAR2(30)	No	Indicates whether tax rate defined for the Retention will be used for Customer Invoices.
PROJECT_CURRENCY_CODE	VARCHAR2(15)	No	Project currency code. This value will not be displayed in the form for release 11.5. Currency code of the set of books will be defaulted.
ALLOW_CROSS_CHARGE_FLAG	VARCHAR2(1)	No	Cross charge allowed? Value is required. Default Value is 'N'. This value can be overridden at any task level.
PROJECT_RATE_DATE	DATE	No	Default project currency rate date (date for accounting currency rate for a given rate type).
PROJECT_RATE_TYPE	VARCHAR2(30)	No	Default project currency rate type (e.g., Spot, Corporate).
CC_PROCESS_LABOR_FLAG	VARCHAR2(1)	No	Flag that indicates cross charge processing is to be performed for labor transactions charged to the project. Default value for the project template is 'N'. This is defaulted to a project from the project template.

Name	Type	required?	Description
LABOR_TP_SCHEDULE_ID	NUMBER	No	Identifier for transfer price schedule for cross charged labor transactions. This is defaulted to a project from the project template. If cc_process_labor_flag is set to 'Y', this field is required.
LABOR_TP_FIXED_DATE	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for labor transactions. This is defaulted to a project from the project template. This value for the project is default for the task fixed date. If cc_process_labor flag is set to 'Y', this field is required.
CC_PROCESS_NL_FLAG	VARCHAR2(1)	No	Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Defaulted value for the project template is 'N'. This is defaulted to a project from the project template.
NL_TP_SCHEDULE_ID	NUMBER	No	Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If cc_process_nl_labor flag is set to 'Y', this field is required.
NL_TP_FIXED_DATE	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for non-labor transactions. This is defaulted to a project from the project template. If cc_process_nl_flag is set to'Y', this field is required.
CC_TAX_TASK_ID	NUMBER	No	Identifier of the task to which intercompany tax items on the intercompany AP invoice are charged.
P_ROLE_LIST_ID	NUMBER(15)	No	Identifier of the role list, a list of allowable roles that are displayed when team members are assigned
P_WORK_TYPE_ID	NUMBER(15)	No	Work type identifier. Work types are predefined types of work. For example, Vacation, Training, and Administration.

Name	Type	required?	Description
P_CALENDAR_ID	NUMBER(15)	No	Calendar identifier. A calendar specifies exceptions such as public holidays.
P_LOCATION_ID	NUMBER(15)	No	Identifier of the project work site location
P_PROBABILITY_MEMBER_ID	NUMBER(15)	No	Identifier of the probability member. Project probability, the likelihood that a project will be approved, is used as a weighting average for reporting.
P_PROJECT_VALUE	NUMBER	No	The opportunity value converted to the project functional currency
P_EXPECTED_APPROVAL_DATE	DATE	No	The expected date of the project approval (for information purposes only)
P_INITIAL_TEAM_TEMPLATE_ID	NUMBER(15)	No	The team template that you want to add to a new project
P_JOB_BILL_RATE_SCHEDULE_ID	NUMBER	No	The identifier of the job-based bill rate schedule for the project
P_EMP_BILL_RATE_SCHEDULE_ID	NUMBER	No	The identifier of the employee-based bill tate schedule for the project
P_COMPETENCE_MATCH_WT	NUMBER	No	The weighting value for competence match, used to calculate the score
P_AVAILABILITY_MATCH_WT	NUMBER	No	The weighting value for availability match, used to calculate the score
P_JOB_LEVEL_MATCH_WT	NUMBER	No	The weighting value for job-level match, used to calculate the score
P_ENABLE_AUTOMATED_SEARCH	VARCHAR2(1)	No	Flag that indicates whether automated candidate nomination is used for the requirements on a project
P_SEARCH_MIN_AVAILABILITY	NUMBER	No	The minimum required availability for a resource to be returned in the search result
P_SEARCH_ORG_HIER_ID	NUMBER(15)	No	Organization hierarchy for searches
P_SEARCH_STARTING_ORG_ID	NUMBER(15)	No	Starting organization for searches
P_SEARCH_COUNTRY_CODE	VARCHAR2(2)	No	Country for searches
P_MIN_CAND_SCORE_REQD_FOR_NOM	NUMBER	No	Minimum score required for a resource to be nominated as candidate on a requirement

Name	Type	required?	Description
P_NON_LAB_STD_BILL_RT_SCH_ID	NUMBER(15)	No	Identifier of the non-labor standard bill rate schedule
P_INVPROC_CURRENCY_TYPE	VARCHAR2(30)	No	Invoice processing currency code
P_REVPROC_CURRENCY_CODE	VARCHAR2(15)	No	Revenue processing currency code in the project functional currency
P_PROJECT_BIL_RATE_DATE_CODE	VARCHAR2(30)	No	Exchange rate date type for converting customer billing amounts from bill transaction currency or funding currency to project currency
P_PROJECT_BIL_RATE_TYPE	VARCHAR2(30)	No	Exchange rate type for converting customer billing amounts from bill transaction currency or funding currency to project currency
P_PROJECT_BIL_RATE_DATE	DATE	No	Exchange rate date for converting customer billing amounts from bill transaction currency or funding currency to project currency, if the rate date type is Fixed.
P_PROJECT_BIL_EXCHANGE_RATE	NUMBER	No	Exchange rate for conversion from bill transaction currency or funding currency to project currency if the rate type is User
P_PROJFUNC_CURRENCY_CODE	VARCHAR2(15)	No	Project functional currency. The default value is the value entered for the associated set of books.
P_PROJFUNC_BIL_RATE_DATE_CODE	VARCHAR2(30)	No	Exchange rate date type for converting customer billing amounts from bill transaction currency or funding currency to project functional currency
P_PROJFUNC_BIL_RATE_TYPE	VARCHAR2(30)	No	Exchange rate type for converting customer billing amounts from bill transaction currency or funding currency to project functional currency
P_PROJFUNC_BIL_RATE_DATE	DATE	No	Exchange rate date for converting customer billing amounts from bill transaction currency or funding currency to project functional currency if the rate date type is Fixed
P_PROJFUNC_BIL_EXCHANGE_RATE	NUMBER	No	Exchange rate for conversion from bill transaction currency or funding currency to project functional currency if the rate type is User

Name	Type	required?	Description
P_FUNDING_RATE_DATE_CODE	VARCHAR2(30)	No	Exchange rate date type for conversting customer billing amounts from bill transaction currency to funding currency
P_FUNDING_RATE_TYPE	VARCHAR2(30)	No	Exchange rate type for converting customer billing amounts from bill transaction currency to funding currency
P_FUNDING_RATE_DATE	DATE	No	Exchange rate date for converting customer billing amounts from bill transaction currency to funding currency if rate date type is Fixed
P_FUNDING_EXCHANGE_RATE	NUMBER	No	Exchange rate for conversion from bill transaction currency to project or functional currency if rate type is User
P_BASELINE_FUNDING_FLAG	VARCHAR2(1)	No	Flag that indicates whether the funding can be baselined without a revenue budget
P_PROJFUNC_COST_RATE_TYPE	VARCHAR2(30)	No	Default value for the project functional cost rate
P_PROJFUNC_COST_RATE_DATE	DATE	No	Default value for the project functional cost rate date
P_INV_BY_BILL_TRANS_CURR_FLAG	VARCHAR2(1)	No	Flag that indicates whether invoicing is by bill transaction currency for the project
P_MULTI_CURRENCY_BILLING_FLAG	VARCHAR2(1)	No	Flag that indicates if multi-currency billing is allowed for the project
P_ASSIGN_PRECEDES_TASK	VARCHAR2(1)	No	Flag that indicatesd if assignment level attributes override task level attributes
P_PRIORITY_CODE	VARCHAR2(30)	No	The code identifying the priority of the project
P_RETN_BILLING_INV_FORMAT_ID	NUMBER(15)	No	The identifier of the retention billing invoice format
P_RETN_ACCOUNTING_FLAG	VARCHAR2(1)	No	Flag that indicates whether retention accounting is enabled for the project
P_ADV_ACTION_SET_ID	NUMBER(15)	No	Flag that indicates the default advertisement action set of the project or project template
P_START_ADV_ACTION_SET_FLAG	VARCHAR2(1)	No	Flag that indicates whether the advertisement action set will start immediately after a requirement is created

Name	Type	required?	Description
P_REVALUATE_FUNDING_FLAG	VARCHAR2(1)	No	Flag that indicates whether the funding has to be revaluated
P_INCLUDE_GAINS_LOSSES_FLAG	VARCHAR2(1)	No	Flag that indicates whether gains and losses to be included in project revenue
P_TARGET_START_DATE	DATE	No	The target start date for the project
P_TARGET_FINISH_DATE	DATE	No	The target finish date for the project
P_BASELINE_START_DATE	DATE	No	The baseline start date of the project
P_SELINE_FINISH_DATE	DATE	No	The baseline finish date of the project
P_SCHEDULED_AS_OF_DATE	DATE	No	The publish date for the scheduled start and finish dates for the project
P_BASELINE_AS_OF_DATE	DATE	No	The baseline date for the baseline start and finish dates for the project
P_LABOR_DISC_REASON_CODE	VARCHAR2(30)	No	Reason code for labor discount
P_NON_LABOR_DISC_REASON_CODE	VARCHAR2(30)	No	Reason code for non-labor discount
P_SECURITY_LEVEL	NUMBER	No	Indicates whether a project is public or private. Zero (0) indicates that the project is private. 100 indicates that the project is public.
P_ACTUAL_AS_OF_DATE	DATE	No	The publish date for the project actual start and actual finish dates
P_SCHEDULED_DURATION	NUMBER	No	Duration from the scheduled start date to the scheduled finish date using the project work calendar
P_BASELINE_DURATION	NUMBER	No	Duration from the baseline start date to the baseline finish date using the project work calendar
P_ACTUAL_DURATION	NUMBER	No	Duration from the actual start date to the actual finish date using the project work calendar
P_LONG_NAME	VARCHAR2(240)	No	Project long name
P_BTC_COST_BASE_REV_CODE	VARCHAR2(90)	No	Bill transaction currency for cost-based revenue
P_ASSET_ALLOCATION_METHOD	VARCHAR2(30)	No	The method used to allocate indirect and common costs across the assets assigned to a grouping level

Name	Type	required?	Description
P_CAPITAL_EVENT_PROCESSING	VARCHAR2(30)	No	The capital event processing method, used to determine when cost and assets are grouped for capitalization or retirement adjustment processing
P_CINT_RATE_SCH_ID	NUMBER(15)	No	Identifier of the capital interest rate schedule
P_CINT_ELIGIBLE_FLAG	VARCHAR2(1)	No	Flag that indicates whether the project is eligible for capitalized interest
P_CINT_STOP_DATE	DATE	No	Stop date for capital interest calculation

PROJECT_OUT_REC_TYPE Datatype

The following table shows the PROJECT_OUT_REC_TYPE datatype.

Name	Type	Req?	Description
PA_PROJECT_ID	NUMBER(15)		The reference code that uniquely identifies the project in Oracle Projects
PA_PROJECT_NUMBER	VARCHAR2(25)		The number that uniquely identifies the project in Oracle Projects
RETURN_STATUS	VARCHAR2(1)		API standard

PROJECT_ROLE_TBL_TYPE Datatype

The following table shows the PROJECT_ROLE_TBL_TYPE datatype.

Name	Type	Req?	Description
PERSON_ID	NUMBER(9)	Based on template setup	The identification code of the employee that manages or administers the project
PROJECT_ROLE_TYPE	VARCHAR2(20)	Yes, if PERSON_ID is not NULL	The type of role that the project player has on the project
START_DATE	DATE	No. Project start date is the default.	Indicates when this person starts playing this role
END_DATE	DATE	No	Indicates when this person stops playing this role

CLASS_CATEGORY_TBL_TYPE Datatype

The following table shows the CLASS_CATEGORY_TBL_TYPE datatype.

Name	Type	Req?	Description
CLASS_CATEGORY	VARCHAR2(30)	Template (based on template setup)	The class category by which the project is classified
CLASS_CODE	VARCHAR2(30)	Yes (only if CLASS_CATEGORY is not NULL)	The class code that classifies the project
CODE_PERCENTAGE	NUMBER	NO	Class category percentage

TASK_IN_TBL_TYPE Datatype

The following table shows the TASK_IN_TBL_TYPE datatype.

Note: If you are using this datatype to update tasks for an existing project, you must include the entire WBS structure in the correct hierarchy.

Name	Type	req?	Description
PM_TASK_REFERENCE	VARCHAR2(25)	Yes, or PA_TASK_ID is given	The reference code that identifies a project's task in the external system
PA_TASK_ID	NUMBER(15)	For update	The reference code that uniquely identifies a task within a project in Oracle Projects
TASK_NAME	VARCHAR2(20)	Yes	The name that uniquely identifies a task within a project
PA_TASK_NUMBER	VARCHAR2(25)	Yes	The number that identifies the task in Oracle Projects. Intended for systems that maintain a task number in addition to a unique task_reference.
TASK_DESCRIPTION	VARCHAR2(250)	No	Description of the task
TASK_START_DATE	DATE	No	The date on which the task starts
TASK_COMPLETION_DATE	DATE	No	The date on which the task is completed
PM_PARENT_TASK_REFERENCE	VARCHAR2(25)	No	The reference code that identifies the task's parent task in the external system
PA_PARENT_TASK_ID	NUMBER	For update	The identification code of the task's parent task in Oracle Projects
ADDRESS_ID	NUMBER	No	The address of one of the customers logically linked to this task

Name	Type	req?	Description
CARRYING_OUT_ORGANIZATION_ID	NUMBER(15)	No	The identification code of the organization responsible for the task work. The task organization defaults to the project organization upon creation of the task.
SERVICE_TYPE_CODE	VARCHAR2(30)	No	The type of work performed on the task
TASK_MANAGER_PERSON_ID	NUMBER(9)	No	The identification code of the employee who manages the task. NOTE: To ensure that the task manager has been defined in Oracle Projects, use a list of values (pa_task_managers_lov_v) to select a task manager's person identification code.
BILLABLE_FLAG	VARCHAR2(1)	No	Default flag for items charged to the task that indicates if the item can accrue revenue (Y or N)
CHARGEABLE_FLAG	VARCHAR2(1)	No	Flag that indicates if expenditure items can be charged to the task. Only lowest tasks are chargeable.
READY_TO_BILL_FLAG	VARCHAR2(1)	No	Flag that indicates whether the task is authorized to be invoiced
READY_TO_DISTRIBUTE_FLAG	VARCHAR2(1)	No	Flag that indicates whether the task is authorized for revenue accrual
LIMIT_TO_TXN_CONTROLS_FLAG	VARCHAR2(1)	No	Flag that indicates that users can charge to the task only those expenditures listed in the task's transaction controls
LABOR_BILL_RATE_ORG_ID	NUMBER(15)	No	The identification code of the organization that owns the labor standard bill rate schedule
LABOR_STD_BILL_RATE_SCHDL	VARCHAR2(20)	No	The labor standard bill rate schedule used to calculate revenue for labor expenditure items charged to the task
LABOR_SCHEDULE_FIXED_DATE	DATE	No	The date used to determine the effective bill rates of the task standard labor bill rate schedule
LABOR_SCHEDULE_DISCOUNT	NUMBER(7,4)	No	The percentage to be discounted from the task standard labor bill rate schedule
NON_LABOR_BILL_RATE_ORG_ID	NUMBER(15)	No	The identification code of the organization that owns the non-labor standard bill rate schedule
NON_LABOR_STD_BILL_RATE_SCHDL	VARCHAR2(30)	No	The non-labor standard bill rate schedule used to calculate revenue for non-labor expenditure items charged to the task

Name	Type	req?	Description
NON_LABOR_SCHEDULE_FIXED_DATE	DATE	No	The fixed date used to determine the effective bill rates of the standard non-labor bill rate schedule
NON_LABOR_SCHEDULE_DISCOUNT	NUMBER(7,4)	No	The percentage to be discounted from the task standard non-labor bill rate schedule
LABOR_COST_MULTIPLIER_NAME	VARCHAR2(20)	No	The labor cost multiplier defined for the task of a premium project. The labor cost multiplier is populated for all overtime expenditure items charged to the task.
COST_IND_RATE_SCH_ID	NUMBER(15)	No	The identification code of the default costing burden schedule
REV_IND_RATE_SCH_ID	NUMBER(15)	No	The identification code of the default revenue burden schedule
INV_IND_RATE_SCH_ID	NUMBER(15)	No	The identification code of the default invoice burden schedule
COST_IND_SCH_FIXED_DATE	DATE	No	The scheduled fixed date of the firm costing burden schedule
REV_IND_SCH_FIXED_DATE	DATE	No	The scheduled fixed date of the firm revenue burden schedule
INV_IND_SCH_FIXED_DATE	DATE	No	The scheduled fixed date of the firm invoice burden schedule
LABOR_SCH_TYPE	VARCHAR2(1)	No	The scheduled type of labor expenditure items
NON_LABOR_SCH_TYPE	VARCHAR2(1)	No	The scheduled type of non-labor expenditure items
ACTUAL_START_DATE	DATE	No	The actual start date of the project in the external system
ACTUAL_FINISH_DATE	DATE	No	The actual finish date of the project in the external system
EARLY_START_DATE	DATE	No	The early start date of the project in the external system
EARLY_FINISH_DATE	DATE	No	The early finish date of the project in the external system
LATE_START_DATE	DATE	No	The late start date of the project in the external system
LATE_FINISH_DATE	DATE	No	The late finish date of the project in the external system
SCHEDULED_START_DATE	DATE	No	The scheduled start date of the project in the external system
SCHEDULED_FINISH_DATE	DATE	No	The scheduled finish date of the project in the external system
ALLOW_CROSS_CHARGE_FLAG	VARCHAR2(1)	No	Cross charge allowed? Value is required. Default Value is 'N'. This value can be overridden at any task level.

Name	Type	req?	Description
PROJECT_RATE_DATE	DATE	No	Default project currency rate date (date for accounting currency rate for a given rate type).
PROJECT_RATE_TYPE	VARCHAR2(30)	No	Default project currency rate type (e.g., Spot, Corporate).
CC_PROCESS_LABOR_FLAG	VARCHAR2(1)	No	Flag that indicates cross charge processing is to be performed for labor transactions charged to the project. Default value for the project template is 'N'. This is defaulted to a project from the project template.
LABOR_TP_SCHEDULE_ID	NUMBER	No	Identifier for transfer price schedule for cross charged labor transactions. This is defaulted to a project from the project template. If cc_process_labor_flag is set to 'Y', this field is required.
LABOR_TP_FIXED_DATE	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for labor transactions. This is defaulted to a project from the project template. This value for the project is default for the task fixed date. If cc_process_labor flag is set to 'Y', this field is required.
CC_PROCESS_NL_FLAG	VARCHAR2(1)	No	Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Defaulted value for the project template is 'N'. This is defaulted to a project from the project template.
NL_TP_SCHEDULE_ID	NUMBER	No	Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If cc_process_nl_labor flag is set to 'Y', this field is required.
NL_TP_FIXED_DAT	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for non-labor transactions. This is defaulted to a project from the project template. If cc_process_nl_flag is set to'Y', this field is required.

Name	Type	req?	Description
RECEIVE_PROJECT_INVOICE_FLAG	VARCHAR2(1)	No	Flag that indicates that the task may receive charges from internal suppliers via inter-project billing.
ATTRIBUTE_CATEGORY	VARCHAR2(30)	No	Used by descriptive flexfields
ATTRIBUTE1 through ATTRIBUTE10	VARCHAR2(150)	No	Descriptive flexfield
P_JOB_BILL_RATE_SCHEDULE_ID	NUMBER	No	The identifier of the job-based bill rate schedule for the project
P_EMP_BILL_RATE_SCHEDULE_ID	NUMBER	No	The identifier of the employee-based bill rate schedule for the project
P_TASKFUNC_COST_RATE_TYPE	VARCHAR2(30)	No	The task-level default value for project functional cost rate type
P_TASKFUNC_COST_RATE_DATE	DATE	No	The task-level default value for project functional cost rate date
P_NON_LAB_STD_BILL_RT_SCH_ID	NUMBER(15)	No	Identifier of the non-labor standard bill rate schedule
P_LABOR_DISC_REASON_CODE	VARCHAR2(30)	No	Reason code for labor discount
P_NON_LABOR_DISC_REASON_CODE	VARCHAR2(30)	No	Reason code for non-labor discount
P_LONG_TASK_NAME	VARCHAR2(240)	No	Task long name
P_RETIREMENT_COST_FLAG	VARCHAR2(1)	No	Flag that identifies tasks for retirement cost collection
P_CINT_ELIGIBLE_FLAG	VARCHAR2(1)	No	Flag that indicates whether the project is eligible for capitalized interest
P_CINT_STOP_DATE	DATE	No	Stop date for capital interest calculation
P_REVENUE_ACCRUAL_METHOD	VARCHAR2(30)	No	The revenue accrual method for task
P_INVOICE_METHOD	VARCHAR2(30)	No	The invoice method for the task
P_OBLIGATION_START_DATE	DATE	No	The obligation start date of the workplan version
P_OBLIGATION_FINISH_DATE	DATE	No	The obligation finish date of the workplan version
P_ACTUAL_START_DATE	DATE	No	The actual start date of the workplan version
P_ACTUAL_FINISH_DATE	DATE	No	The actual end date of the workplan version
P_ESTIMATED_START_DATE	DATE	No	The estimated start date of the workplan version
P_ESTIMATED_FINISH_DATE	DATE	No	The estimated finish date of the workplan version
P_EARLY_START_DATE	DATE	No	The early start date of the workplan version

Name	Type	req?	Description
P_EARLY_FINISH_DATE	DATE	No	The early finish date of the workplan version
P_LATE_START_DATE	DATE	No	The late start date of the workplan version
P_LATE_FINISH_DATE	DATE	No	The late finish date of the workplan version
P_MILESTONE_FLAG	VARCHAR2(1)	No	Flag that indicates if the task version is a milestone. This is a task-specific attribute.
P_CRITICAL_FLAG	VARCHAR2(1)	No	Flag that indicates if the task version is part of the critical path. This is a task-specific attribute.
P_WQ_PLANNED_QUANTITY	NUMBER(17)	No	The planned work quantity for the task
P_PLANNED_EFFORT	NUMBER(17)	No	The planned effort for the task

TASK_OUT_TBL_TYPE Datatype

The following table shows the TASK_OUT_TBL_TYPE datatype.

Name	Type	req?	Description
PA_TASK_ID	NUMBER(15)		The reference code that uniquely identifies a task within a project in Oracle Projects
PM_TASK_REFERENCE	VARCHAR2(25)		The reference code that identifies a project's task in the external system
RETURN_STATUS	VARCHAR2(1)		API standard
TASK_VERSION_ID	NUMBER		Task Version ID

Project API Procedure Definitions

This section contains description of the project APIs, including business rules and parameters.

Common Project API Parameters

The following descriptions apply to columns that are used throughout the Project APIs.

PM_PROJECT_REFERENCE

Systems that you use to create projects in Oracle Projects assign a unique number to every project. You can set up Oracle Projects either to generate project numbers automatically or to support manual entry of numbers.

When Oracle Projects is set up for automatic numbering:

- The number generated automatically by Oracle Projects is stored in the column SEGMENT1.
- The number assigned by the external system is stored in the column PM_PROJECT_REFERENCE.

When Oracle Projects is set up for manual numbering, the number assigned by the external system is stored in both SEGMENT1 and PM_PROJECT_REFERENCE.

> **Note:** Oracle Projects windows display only SEGMENT1 as the project number, so you should set up Oracle Projects to support manual numbering if you plan to integrate Oracle Projects with an external system.

Project and Task Start and Finish Dates

Most external systems hold additional start and finish dates for projects and tasks. For information about using a client extension to pass these additional dates (instead of the default Oracle Projects project dates), see Project and Task Date Client Extension: page 8 – 8.

ADD_TASK

ADD_TASK is a PL/SQL procedure used to add new subtasks to a task of a project in Oracle Projects. We replaced the task record type with a parameter with a standard datatype (NUMBER, VARCHAR2, or DATE) for every field in the record type definition so you can call this procedure directly.

Business Rules (task level)

Oracle Projects imposes the following task–level business rules:

- Each new task must have a unique number within a given project. You can use the Check procedure CHECK_UNIQUE_TASK_NUMBER to verify that the new task number does not already exist in your project.
- You cannot create a subtask for any project if the parent task has any of the following attributes:
  - Transaction controls
  - Burden schedule overrides
  - A budget
  - A percentage complete value
  - An asset
  - An expenditure item
  - A purchase order distribution
  - A purchase order requisition
  - An Oracle Payables invoice
  - An Oracle Payables invoice distribution

  Note: You can use the Check procedure CHECK_ADD_SUBTASK_OK to verify that you can add a subtask to a particular parent task.
- For contract projects, you cannot add a subtask to a parent task that has any of the following attributes:
  - Labor cost multiplier
  - Job bill rate override
  - Employee bill rate override
  - Labor multiplier

– Non–labor bill rate override

– Job bill title override

– Job assignment override

Note: You can use the Check procedure CHECK_ADD_SUBTASK_OK to verify that you can add a subtask to a particular parent task.

The following table shows the parameters for ADD_TASK.

Note: Some parameters in this table have "See: TASK_IN_TBL_TYPE" as their description. The descriptions for these parameters are shown in the parameter list for the TASK_IN_TBL_TYPE datatype on page 3 – 15. (The parameter names are identical in TASK_IN_TBL_TYPE, except that they do not begin with "P_".)

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	Code identifying the external system
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	Yes	See the TASK_IN_TBL_TYPE Datatype table on page 3 – 15 for a description of this field. By default, you can pass the same value for both PM_TASK_REFERENCE and PA_TASK_NUMBER.
P_PA_TASK_NUMBER	IN	VARCHAR2(25)	Yes	By default, you can pass the same value for both PM_TASK_REFERENCE and PA_TASK_NUMBER.
P_TASK_NAME	IN	VARCHAR2(20)	Yes	*See: TASK_IN_TBL_TYPE*
P_TASK_DESCRIPTION	IN	VARCHAR2(250)	No	*See: TASK_IN_TBL_TYPE*
P_TASK_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_TASK_COMPLETION_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*

Name	Usage	Type	Req?	Description
P_PM_PARENT_TASK_REFERENCE	IN	VARCHAR2(25)	No	*See: TASK_IN_TBL_TYPE*
P_ADDRESS_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_CARRYING_OUT_ORGANIZATION_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_SERVICE_TYPE_CODE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_TASK_MANAGER_PERSON_ID	IN	NUMBER(9)	No	*See: TASK_IN_TBL_TYPE*
P_BILLABLE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CHARGEABLE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_READY_TO_BILL_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_READY_TO_DISTRIBUTE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_LIMIT_TO_TXN_CONTROLS_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_BILL_RATE_ORG_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_STD_BILL_RATE_SCHDL	IN	VARCHAR2(20)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_SCHEDULE_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_SCHEDULE_DISCOUNT	IN	NUMBER(7,4)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_BILL_RATE_ORG_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_STD_BILL_RATE_SCHDL	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_SCHEDULE_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_SCHEDULE_DISCOUNT	IN	NUMBER(7,4)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_COST_MULTIPLIER_NAME	IN	VARCHAR2(20)	No	*See: TASK_IN_TBL_TYPE*
P_COST_IND_RATE_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_REV_IND_RATE_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_INV_IND_RATE_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_COST_IND_SCH_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_REV_IND_SCH_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_INV_IND_SCH_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_SCH_TYPE	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_SCH_TYPE	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_SCHEDULED_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_SCHEDULED_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*

Name	Usage	Type	Req?	Description
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	*See: TASK_IN_TBL_TYPE*
P_JOB_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_EMP_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_TASKFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_TASKFUNC_COST_RATE_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_NON_LAB_STD_BILL_RT_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_LONG_TASK_NAME	IN	VARCHAR2(240)	No	*See: TASK_IN_TBL_TYPE*
P_RETIREMENT_COST_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CINT_ELIGIBLE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CINT_STOP_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_REVENUE_ACCRUAL_METHOD	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_INVOICE_METHOD	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_OBLIGATION_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_OBLIGATION_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ESTIMATED_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ESTIMATED_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_MILESTONE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CRITICAL_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_WQ_PLANNED_QUANTITY	IN	NUMBER(17)	No	*See: TASK_IN_TBL_TYPE*
P_PLANNED_EFFORT	IN	NUMBER(17)	No	*See: TASK_IN_TBL_TYPE*
P_ALLOW_CROSS_CHARGE_FLAG	IN	VARCHAR2(1)	No	Cross charge allowed? Value is required. Default Value is 'N'. This value can be overridden at any task level.
P_PROJECT_RATE_DATE	IN	DATE	No	Default project currency rate date (date for accounting currency rate for a given rate type).
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Default project currency rate type (e.g., Spot, Corporate).

Name	Usage	Type	Req?	Description
P_CC_PROCESS_LABOR_FLAG	IN	VARCHAR2(1)	No	Flag that indicates cross charge processing is to be performed for labor transactions charged to the project. Default value for the project template is 'N'. This is defaulted to a project from the project template.
P_LABOR_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged labor transactions. This is defaulted to a project from the project template. If cc_process_labor_flag is set to 'Y', this field is required.
P_LABOR_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for labor transactions. This is defaulted to a project from the project template. This value for the project is default for the task fixed date. If cc_process_labor flag is set to 'Y', this field is required.
P_CC_PROCESS_NL_FLAG	IN	VARCHAR2(1)	No	Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Defaulted value for the project template is 'N'. This is defaulted to a project from the project template.
P_NL_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If cc_process_nl_labor flag is set to 'Y', this field is required.
P_NL_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for non-labor transactions. This is defaulted to a project from the project template. If cc_process_nl_flag is set to'Y', this field is required.
P_RECEIVE_PROJECT_INVOICE_FLAG	IN	VARCHAR2(1)	No	Flag that indicates that the task may receive charges from internal suppliers via inter-project billing.
P_PA_PROJECT_ID_OUT	OUT	NUMBER(15)		API standard
P_PA_PROJECT_NUMBER_OUT	OUT	VARCHAR2(25)		API standard
P_TASK_ID	OUT	NUMBER(15)		API standard

CREATE_PROJECT

CREATE_PROJECT is a PL/SQL procedure that creates a project in Oracle Projects using a template or an existing project.

> **Note:** CREATE_PROJECT will not copy the WBS structure to the newly created project when attempting to copy a project or template with tasks.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes: page 2 – 19.

> **Note:** When loading descriptive flexfields using Oracle Projects APIs, if the DFF is not context sensitive, then the parameter ATTRIBUTE_CATEGORY is required to have a value such as 'Global Data Elements'. Otherwise, the APIs do not import rows.

The following table shows the parameters for CREATE_PROJECT.

Name	Usage	TYPE	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_WORKFLOW_STARTED	OUT	VARCHAR2		Shows if a workflow has been started (Y or N)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	Code identifying the external system
P_PROJECT_IN	IN	PROJECT_IN_REC_TYPE	Yes	See: PROJECT_IN_REC_TYPE Datatype table: page 3 – 7
P_PROJECT_OUT	OUT	PROJECT_OUT_REC_TYPE		See the PROJECT_OUT_REC_TYPE Datatype table on page 3 – 14
P_KEY_MEMBERS	IN	PROJECT_ROLE_TBL_TYPE	No	See the PROJECT_ROLE_TBL_TYPE Datatype table on page 3 – 14
P_CLASS_CATEGORIES	IN	CLASS_CATEGORY_TBL_TYPE	No	See the CLASS_CATEGORY_TBL_TYPE Datatype table on page 3 – 15
P_TASKS_IN	IN	TASK_IN_TBL_TYPE	No	See the TASK_IN_TBL_TYPE Datatype table on page 3 – 15
P_TASKS_OUT	OUT	TASK_OUT_TBL_TYPE		See the TASK_OUT_TBL_TYPE Datatype table on page 3 – 20
P_ORG_ROLES	IN	TABLE TYPE	No	Organization roles record type
P_STRUCTUR_IN	IN	TABLE TYPE	No	Structure types. Dfault = "Financial"
P_EXT_ATTR_TBL_IN	IN	TABLE TYPE	No	Extensible attribute table

DELETE_PROJECT

DELETE_PROJECT is a PL/SQL procedure used to delete a project and its tasks from Oracle Projects.

Business Rule (project level)

You cannot delete a project if any of these items exist:

- Event
- Expenditure item
- Purchase order distribution
- Purchase order requisition
- Supplier invoice
- Invoice distribution
- Funding
- Budget
- Commitment transaction
- Compensation rule set
- Reference from other project

Business Rule (task level)

You cannot delete a project if any of its tasks cannot be deleted. Use the Check procedure CHECK_DELETE_TASK_OK to see if you can delete a certain task. You cannot delete a task if any of the following exists:

- Event at top task
- Funding at top task
- Budget at top task
- Expenditure item at lowest task
- Purchase order line at lowest task
- Requisition line at lowest task
- Supplier invoice (Oracle Payables invoice) at lowest task
- Budget at lowest task

The following table shows the DELETE_PROJECT parameters.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default =F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default =F)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	Code identifying the external system
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects

DELETE_TASK

DELETE_TASK is a PL/SQL procedure used to delete tasks of a project in Oracle Projects.

Business Rules (task level)

Oracle Projects imposes the following business rules.

Cascaded Task Deletion

The following rules apply to cascaded task deletion. In cascaded task deletion, when a task is deleted, all of its subtasks are also deleted.

You can delete a top task only if the task satisfies Rules 1 through 8:

1. No top task event, such as revenue or billing, exists
2. No top task funding exists
3. No top task budget exists

You can delete a mid or lowest task if the task satisfies Rules 4 through 8 (for a mid task, the rules relate to the lowest tasks below that mid task):

4. No lowest task expenditure item exists

5. No lowest task purchase order line exists
6. No lowest task requisition line exists
7. No lowest task supplier invoice exists
8. No lowest task budget exists

Non–Cascaded Task Deletion

The following business rules apply to non–cascaded task deletion. In non–cascaded task deletion, deleting a task deletes only that task, and moves all subtasks below it up one level in the project's work breakdown structure.

- You can delete a mid task at all times.
- You can delete a top task if it satisfies Rules 1 through 3 for cascaded task deletion.
- You can delete a lowest task if it satisfies Rules 4 through 8 for cascaded task deletion.

The following table shows the DELETE_TASK parameters.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	Code identifying the external system
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the task in the external system
P_PA_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_CASCADED_DELETE_FLAG	IN	VARCHAR2(1)	No	When 'Y' is passed, this task and all its subtasks are deleted (default = 'N')

Name	Usage	Type	Req?	Description
P_PROJECT_ID	OUT	NUMBER(15)		API standard
P_TASK_ID	OUT	NUMBER(15)		API standard
P_TASK_VERSION_ID	IN	NUMBER	No	Task version ID
P_STRUCTURE_TYPE	IN	VARCHAR2	No	Structure type (default = Financial)

UPDATE_PROJECT

UPDATE_PROJECT is a PL/SQL procedure that updates project and task information from your external system to Oracle Projects to reflect changes you have made in the external system.

UPDATE_PROJECT uses composite datatypes. For more information about composite datatypes, see APIs That Use Composite Datatypes: page 2 – 19.

Oracle Projects imposes project– and task–level business rules that restrict the changes you can make to project and task information. To ensure that Oracle Projects accepts all the project or task changes you make in your external system, review the following rules before you make changes in your external system. You can also use the check procedures: page 3 – 61 to identify the types of changes that Oracle Projects supports.

Business Rules (Project Level)

Oracle Projects imposes the following business rules.

Project Numbers, Project Names, Project Types, and Project Organizaitons

The following rules apply to project numbers, names, types, and organizations:

- **Project Number** : You cannot change a project number if expenditure items or invoices have been charged to the project.
  - New project numbers must be unique within Oracle Projects. Use CHECK_UNIQUE_PROJECT_REFERENCE (a Check procedure) to verify that the new project number is unique.
  - If you use an external system to create original project plans, choose manual project numbering. Numbers

generated automatically by external systems may not be unique in Oracle Projects and will be replaced by new project numbers generated by Oracle Projects.

- **Project Name:** The new project name must be unique.
- **Project Type:** You cannot change the project type (indirect, capital, or contract) of a project.
- **Project Organization:** You cannot change the project organization if cost distribution lines, draft revenue, or draft invoices have been charged against the project.

Team Members and Customers

The rules for project team members and customers are shown in the following table:

Entity or Topic	Rule
Project manager	A project can have only one active project manager.
New project manager	If you assign a new project manager to an existing project, the default start date for the new project manager is the system date. The default end date for the current project manager is the previous day.
Team members	A project can have any number of team members other than the project manager.
Team member start date	If the start date of a team member other than a project manager is not passed or passed as NULL, the start day is derived from the project start date. When project_start_date is NULL, the default start date for the key member is NULL.
Team roles during different periods	UPDATE_PROJECT does not support a person performing the same role (other than project manager) for a given project during different periods.
Primary customer	A project can have only one primary customer.

Table 3 – 4 UPDATE_PROJECT rules for team members and customers (Page 1 of 1)

Rules for Project Start and End Dates

The rules for project start and end dates are shown in the following table:

- Project start and completion dates must include the first task start date and the last task completion date for all tasks included in the project.
- You can leave both the start and completion dates or just the completion date blank; however, you must enter a start date if you want to enter a completion date.
- If you change the project status to Closed, then the default completion date is the system date. If you subsequently reopen the project, the default completion date is NULL.
- A NULL value for any of the project fields listed below results in an error message in Oracle Projects. Oracle Projects ignores incoming NULL values for these fields and retains their original values.
  - PROJECT_STATUS
  - PUBLIC_SECTOR_FLAG
  - PROJECT_NUMBER
  - PROJECT_NAME
  - CARRYING_OUT_ORGANIZATION_ID
  - DISTRIBUTION_RULE for a contract project. (A NULL value for this field raises an error.)

Business Rules (Task Level)

Oracle Projects imposes the following business rules at the task level..

Order in Which Information is Shared

The following rule applies to the order in which task information is shared between your external system and Oracle Projects:

- You must interface parent tasks to Oracle Projects before you can interface the related child tasks.

Task Numbers, Identification Codes, and Organizations

The following rules apply to task numbers, identification codes, and organizations:

- New task numbers must be unique within a project. Use the Check procedure CHECK_UNIQUE_TASK_NUMBER to verify that a new task number is unique in Oracle Projects.

- If the external system pushes both the TASK_ID and the PM_TASK_REFERENCE to Oracle Projects, Oracle Projects uses the TASK_ID to identify the task and updates PM_TASK_REFERENCE with the incoming value (if different).
- You cannot change a task number if any of the following items have been charged against the task:
  - Expenditure items
  - Purchase order distributions
  - Purchase order requisition distributions
  - Supplier invoices
  - Supplier invoice distributions

  Note: Use the Check procedure CHECK_TASK_NUMBER_CHANGE_OK to verify if Oracle Projects allows you to change the number of a certain task.
- You cannot change a task organization if any of the following items have been charged against the task:
  - Cost distribution lines
  - Revenue distribution lines
  - Draft invoices

Task Start and Finish Dates

The following rules apply to task start and finish dates:

- A task start date must occur:
  - After the parent task start date
  - Before the start date of any subtasks
  - Between the project start and completion dates
- Each task with a completion date must also have a start date.
- A task completion date must occur before the project completion date.

Moving a Task in the WBS

Thes following rules apply to moving a task within a project's work breakdown structure (WBS):

- Because billing, budgeting, and creating capital assets are driven from top tasks, you can move a subtask only if its new parent task belongs to the same top task.
- You cannot change a top task to a subtask.
- You cannot change a subtask to a top task.

Task Attributes

These rules apply to task attributes:

- You cannot change any of the following task attributes to NULL:
  - TASK_NAME
  - PM_TASK_REFERENCE
  - TASK_NUMBER
  - READY_TO_BILL_FLAG
  - READY_TO_DISTRIBUTE_FLAG
  - CARRYING_OUT_ORGANIZATION_ID
  - SERVICE_TYPE_CODE
- You can change the following task attributes without restriction:
  - Task manager
  - Description
  - Other flags (not mentioned previously)
  - Labor and non–labor data
  - Schedules and rates

Parameters for UPDATE_PROJECT

The following table shows the parameters for UPDATE_PROJECT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

Name	Usage	Type	Req?	Description
P_WORKFLOW_STARTED	OUT	VARCHAR2(1)		Shows if a workflow has been started (Y or N)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	Code identifying the external system
P_PROJECT_IN	IN	PROJECT_IN_REC_TYPE	Yes	See the PROJECT_IN_REC_TYPE Datatype table on page 3 – 7
P_PROJECT_OUT	OUT	PROJECT_OUT_REC_TYPE		See the PROJECT_OUT_REC_TYPE Datatype table on page 3 – 14
P_KEY_MEMBERS	IN	PROJECT_ROLE_TBL_TYPE	No	See the PROJECT_ROLE_TBL_TYPE Datatype table on page 3 – 14
P_CLASS_CATEGORIES	IN	CLASS_CATEGORY_TBL_TYPE	No	See the CLASS_CATEGORY_TBL_TYPE Datatype table on page 3 – 15
P_TASKS_IN	IN	TASK_IN_TBL_TYPE	No	See the TASK_IN_tbl_type Datatype table on page 3 – 15. NOTE: If you are using this parameter to update tasks for an existing project, you must include the entire WBS structure in the correct hierarchy.
P_TASKS_OUT	OUT	TASK_OUT_TBL_TYPE		See the TASK_OUT_tbl_type Datatype table on page 3 – 20
P_ORG_ROLES	IN	RECORD TYPE	No	Organization roles record type
P_STRUCTURE_IN	IN	RECORD TYPE	No	Structure record type
P_EXT_ATTR_TBL_IN	IN	RECORD TYPE	No	Extensible attributes record type

UPDATE_TASK

UPDATE_TASK is a PL/SQL procedure used to update existing tasks of a project in Oracle Projects. We replaced the task record type with a parameter that uses a standard datatype (VARCHAR2, NUMBER, and DATE) for every field in the record type definition so you can call this procedure directly.

Business Rules (task level)

Oracle Projects imposes the following business rules.

This rule applies to the order in which task information is shared between your external system and Oracle Projects:

- You must interface the definitions of parent tasks to Oracle Projects before you can interface the definitions of the related child tasks.

The following rules apply to task numbers, identification codes, and organizations:

- A new task number must be unique within a project. (You can use the Check procedure CHECK_UNIQUE_TASK_NUMBER to verify whether your new task number is unique in Oracle Projects.)
- If the external system pushes both the TASK_ID and the PM_TASK_REFERENCE to Oracle Projects, Oracle Projects uses the TASK_ID to identify the task and updates PM_TASK_REFERENCE with the incoming value (if different).
- You cannot change a task number if any of the following items have been charged against the task:
  - Expenditure items
  - Purchase order distributions
  - Purchase order requisition distributions
  - Supplier invoices
  - Supplier invoice distributions

  Note: You can use the Check procedure CHECK_TASK_NUMBER_CHANGE_OK to verify whether Oracle Projects will allow you to change the number of a certain task.
- You cannot change a task organization if any of the following items have been charged against the task:
  - Cost distribution lines
  - Revenue distribution lines
  - Draft invoices

The following rules apply to task start and completion dates:

- A task start date must occur:

  - After the parent task start date
  - Before the start date of any subtasks
  - Between the project start and completion dates
- Each task with a completion date must also have a start date.
- A task completion date must occur before the project completion date.

The following rules apply to moving a task within a project's work breakdown structure (WBS).

- You can move a subtask as long as its new parent task belongs to the same top task, because billing, budgeting, and creating capital assets are driven from top tasks.
- You cannot change a top task to a subtask.
- You cannot change a subtask to a top task.

The following rules apply to changing task fields and attributes:

- You cannot update task fields with a NULL value. Only a field with a valid NOT NULL value will be updated.
- You cannot change any of the following task fields to NULL:
  - TASK_NAME
  - PM_TASK_REFERENCE
  - TASK_NUMBER
  - READY_TO_BILL_FLAG
  - READY_TO_DISTRIBUTE_FLAG
  - CARRYING_OUT_ORGANIZATION_ID
  - SERVICE_TYPE_CODE
- You can change the following task attributes without restriction:
  - Task manager
  - Description
  - Other flags not mentioned previously
  - Labor and non–labor data
  - Schedules and rates

The following table shows the parameters for UPDATE_TASK.

Note: Some parameters in this table have "See: TASK_IN_TBL_TYPE" as their description. The descriptions for these parameters are shown in the parameter list for the TASK_IN_TBL_TYPE datatype on page 3 – 15. (The parameter names are identical in TASK_IN_TBL_TYPE, except that they do not begin with "P_".)

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	Code identifying the external system
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	Yes	*See: TASK_IN_TBL_TYPE*
P_PA_TASK_ID	IN	NUMBER(15)	Yes	*See: TASK_IN_TBL_TYPE*
P_TASK_NAME	IN	VARCHAR2(20)	Yes	*See: TASK_IN_TBL_TYPE*
P_PA_TASK_NUMBER	IN	VARCHAR2(25)	Yes	*See: TASK_IN_TBL_TYPE*
P_TASK_DESCRIPTION	IN	VARCHAR2(250)	No	*See: TASK_IN_TBL_TYPE*
P_TASK_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_TASK_COMPLETION_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_PM_PARENT_TASK_REFERENCE	IN	VARCHAR2(25)	No	*See: TASK_IN_TBL_TYPE*
P_PA_PARENT_TASK_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_ADDRESS_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_CARRYING_OUT_ORGANIZATION_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_SERVICE_TYPE_CODE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_TASK_MANAGER_PERSON_ID	IN	NUMBER(9)	No	*See: TASK_IN_TBL_TYPE*
P_BILLABLE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CHARGEABLE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_READY_TO_BILL_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_READY_TO_DISTRIBUTE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_LIMIT_TO_TXN_CONTROLS_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_BILL_RATE_ORG_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*

Name	Usage	Type	Req?	Description
P_LABOR_STD_BILL_RATE_SCHDL	IN	VARCHAR2(20)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_SCHEDULE_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_SCHEDULE_DISCOUNT	IN	NUMBER(7,4)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_BILL_RATE_ORG_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_STD_BILL_RATE_SCHDL	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_SCHEDULE_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_SCHEDULE_DISCOUNT	IN	NUMBER(7,4)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_COST_MULTIPLIER_NAME	IN	VARCHAR2(20)	No	*See: TASK_IN_TBL_TYPE*
P_COST_IND_RATE_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_REV_IND_RATE_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_INV_IND_RATE_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_COST_IND_SCH_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_REV_IND_SCH_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_INV_IND_SCH_FIXED_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_SCH_TYPE	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_SCH_TYPE	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_SCHEDULED_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_SCHEDULED_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_JOB_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_EMP_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	*See: TASK_IN_TBL_TYPE*
P_TASKFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_TASKFUNC_COST_RATE_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_NON_LAB_STD_BILL_RT_SCH_ID	IN	NUMBER(15)	No	*See: TASK_IN_TBL_TYPE*
P_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_NON_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_LONG_TASK_NAME	IN	VARCHAR2(240)	No	*See: TASK_IN_TBL_TYPE*
P_RETIREMENT_COST_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CINT_ELIGIBLE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CINT_STOP_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*

Name	Usage	Type	Req?	Description
P_REVENUE_ACCRUAL_METHOD	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_INVOICE_METHOD	IN	VARCHAR2(30)	No	*See: TASK_IN_TBL_TYPE*
P_OBLIGATION_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_OBLIGATION_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ACTUAL_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ESTIMATED_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_ESTIMATED_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_EARLY_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_START_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_LATE_FINISH_DATE	IN	DATE	No	*See: TASK_IN_TBL_TYPE*
P_MILESTONE_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_CRITICAL_FLAG	IN	VARCHAR2(1)	No	*See: TASK_IN_TBL_TYPE*
P_WQ_PLANNED_QUANTITY	IN	NUMBER(17)	No	*See: TASK_IN_TBL_TYPE*
P_PLANNED_EFFORT	IN	NUMBER(17)	No	*See: TASK_IN_TBL_TYPE*
P_ALLOW_CROSS_CHARGE_FLAG	IN	VARCHAR2(1)	No	Cross charge allowed? Value is required. Default Value is 'N'. This value can be overridden at any task level.
P_PROJECT_RATE_DATE	IN	DATE	No	Default project currency rate date (date for accounting currency rate for a given rate type).
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Default project currency rate type (e.g., Spot, Corporate).
P_CC_PROCESS_LABOR_FLAG	IN	VARCHAR2(1)	No	Flag that indicates cross charge processing is to be performed for labor transactions charged to the project. Default value for the project template is 'N'. This is defaulted to a project from the project template.
P_LABOR_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged labor transactions. This is defaulted to a project from the project template. If cc_process_labor_flag is set to 'Y', this field is required.

Name	Usage	Type	Req?	Description
P_LABOR_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for labor transactions. This is defaulted to a project from the project template. This value for the project is default for the task fixed date. If cc_process_labor flag is set to 'Y', this field is required.
P_CC_PROCESS_NL_FLAG	IN	VARCHAR2(1)	No	Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Defaulted value for the project template is 'N'. This is defaulted to a project from the project template.
P_NL_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If cc_process_nl_labor flag is set to 'Y', this field is required.
P_NL_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for non-labor transactions. This is defaulted to a project from the project template. If cc_process_nl_flag is set to'Y', this field is required.
P_RECEIVE_PROJECT_INVOICE_FLAG	IN	VARCHAR2(1)	No	Flag that indicates that the task may recive charges from internal suppliers via inter-project billing.
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	
P_OUT_PA_TASK_ID	OUT	NUMBER(15)		API standard
P_OUT_PM_TASK_REFERENCE	OUT	VARCHAR2(25)		API standard

CLEAR_PROJECT

CLEAR_PROJECT is a Load–Execute–Fetch procedure used to clear the global data structures set up during the Load process.

EXECUTE_CREATE_PROJECT

EXECUTE_CREATE_PROJECT is a Load–Execute–Fetch procedure used to create a project and its tasks using the data stored in the global tables during the Load process.

To populate a project with user–defined attributes, this procedure calls the user–defined attribute procedures. For more information, see: User–Defined Attribute APIS: page 3 – 84.

The following table shows the parameters for EXECUTE_CREATE_PROJECT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	
P_COMMIT	IN	VARCHAR2(1)	No	Default = 'F'
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Default = 'F'
P_MSG_COUNT	OUT	NUMBER		
P_MSG_DATA	OUT	VARCHAR2(2000)		
P_RETURN_STATUS	OUT	VARCHAR2(1)		
P_WORKFLOW_STARTED	OUT	VARCHAR2(1)		Shows if a workflow has been started (Y or N)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	
P_PA_PROJECT_ID	OUT	NUMBER		
P_PA_PROJECT_NUMBER	OUT	VARCHAR2(25)		

EXECUTE_UPDATE_PROJECT

EXECUTE_UPDATE_PROJECT is a Load–Execute–Fetch procedure used to update an existing project, including changing or adding project data, adding new tasks, and updating existing tasks. This API does not delete tasks; rather, it uses the data stored in the global tables during the Load process.

To update the user–defined attributes in a project, this procedure calls the user–defined attribute procedures. For more information, see: User–Defined Attribute APIS: page 3 – 84.

The following table shows the parameters for EXECUTE_UPDATE_PROJECT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	
P_COMMIT	IN	VARCHAR2(1)	No	Default = 'F'

Name	Usage	Type	Req?	Description
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Default = 'F'
P_MSG_COUNT	OUT	NUMBER		
P_MSG_DATA	OUT	VARCHAR2(2000)		
P_RETURN_STATUS	OUT	VARCHAR2(1)		
P_WORKFLOW_STARTED	OUT	VARCHAR2(1)		Shows if a workflow has been started (Y or N)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	

FETCH_TASK

FETCH_TASK is a Load–Execute–Fetch procedure used to fetch output parameters related to tasks.

The following table shows the parameters for FETCH_TASK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Default = 'F'
P_RETURN_STATUS	OUT	VARCHAR2(1)		
P_TASK_INDEX	IN	NUMBER	Yes	
P_PA_TASK_ID	OUT	NUMBER(15)		
P_PM_TASK_REFERENCE	OUT	VARCHAR2(25)		
P_TASK_RETURN_STATUS	OUT	VARCHAR2(1)		

FETCH_TASKS

FETCH_TASKS is a wrapper for FETCH_TASK to handle multiple calls to FETCH_TASK.

The following table shows the parameters for FETCH_TASKS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Default = 'F'
P_RETURN_STATUS	OUT	VARCHAR2(1)		
P_TASK_INDEX	IN	NUMBER	Yes	
P_PA_TASK_ID	OUT	NUMBER(15)		

Name	Usage	Type	Req?	Description
P_PM_TASK_REFERENCE	OUT	VARCHAR2(25)		
P_TASK_RETURN_STATUS	OUT	VARCHAR2(1)		

INIT_PROJECT

INIT_PROJECT is a Load–Execute–Fetch procedure used to set up the global data structures. Other Load–Execute–Fetch procedures use the structures to create a new project in Oracle Projects.

LOAD_CLASS_CATEGORY

LOAD_CLASS_CATEGORY is a Load–Execute–Fetch procedure used to load class categories to a global PL/SQL table.

The following table shows the parameters for LOAD_CLASS_CATEGORY.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_CLASS_CATEGORY	IN	VARCHAR2(30)	Depends on template setup	
P_CLASS_CODE	IN	VARCHAR2(30)	Yes, If p_class_category is not NULL	
P_CODE_PERCENTAGE	IN	NUMBER	No	Class category percentage

LOAD_KEY_MEMBER

LOAD_KEY_MEMBER is a Load–Execute–Fetch procedure used to load key members to a global PL/SQL table.

The following table shows the parameters for LOAD_KEY_MEMBER.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)

Name	Usage	Type	Req?	Description
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PERSON_ID	IN	NUMBER(9)	Depends on template setup	
P_PROJECT_ROLE_TYPE	IN	VARCHAR2(20)	Yes, If P_PERSON_ID is not NULL	
P_START_DATE	IN	DATE	No	Default = sysdate
P_END_DATE	IN	DATE	No	

LOAD_PROJECT

LOAD_PROJECT is a Load–Execute–Fetch procedure used to load a project to a global PL/SQL record.

The following table shows the parameters for LOAD_PROJECT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Default = 'F'
P_RETURN_STATUS	OUT	VARCHAR2(1)	No	
P_PA_PROJECT_ID	IN	NUMBER	No, used for update only	
P_PA_PROJECT_NUMBER	IN	VARCHAR2(25)	No, used for update only	
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	
P_PROJECT_NAME	IN	VARCHAR2(30)	Yes	
P_CREATED_FROM_PROJECT_ID	IN	NUMBER(15)	Yes	
P_CARRYING_OUT_ORGANIZATION_ID	IN	NUMBER(15)	Depends on template setup	
P_PUBLIC_SECTOR_FLAG	IN	VARCHAR2(1)	Depends on template setup	
P_PROJECT_STATUS_CODE	IN	VARCHAR2(30)	Depends on template setup	

Name	Usage	Type	Req?	Description
P_DESCRIPTION	IN	VARCHAR2(250)	Depends on template setup	
P_START_DATE	IN	DATE	Depends on template setup	
P_COMPLETION_DATE	IN	DATE	Depends on template setup	
P_DISTRIBUTION_RULE	IN	VARCHAR2(30)	Depends on template setup	
P_CUSTOMER_ID	IN	NUMBER(15)	Depends on template setup	
P_PROJECT_RELATIONSHIP_CODE	IN	VARCHAR2(30)	Depends on template setup	
P_ACTUAL_START_DATE	IN	DATE	No	
P_ACTUAL_FINISH_DATE	IN	DATE	No	
P_EARLY_START_DATE	IN	DATE	No	
P_EARLY_FINISH_DATE	IN	DATE	No	
P_LATE_START_DATE	IN	DATE	No	
P_LATE_FINISH_DATE	IN	DATE	No	
P_SCHEDULED_START_DATE	IN	DATE	No	
P_SCHEDULED_FINISH_DATE	IN	DATE	No	
P_OUTPUT_TAX_CODE	IN	VARCHAR2(30)	No	Indicates whether tax rate defined for the Project will be used for Customer Invoices.
P_RETENTION_TAX_CODE	IN	VARCHAR2(30)	No	Indicates whether tax rate defined for the Retention will be used for Customer Invoices.
P_PROJECT_CURRENCY_CODE	IN	VARCHAR2(15)	No	Project currency code. This value will not be displayed in the form for release 11.5. Currency code of the set of books will be defaulted.

Name	Usage	Type	Req?	Description
P_ALLOW_CROSS_CHARGE_FLAG	IN	VARCHAR2(1)	No	Cross charge allowed? Value is required. Default Value is 'N'. This value can be overridden at any task level.
P_PROJECT_RATE_DATE	IN	DATE	No	Default project currency rate date (date for accounting currency rate for a given rate type).
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Default project currency rate type (e.g., Spot, Corporate).
P_CC_PROCESS_LABOR_FLAG	IN	VARCHAR2(1)	No	Flag that indicates cross charge processing is to be performed for labor transactions charged to the project. Default value for the project template is 'N'. This is defaulted to a project from the project template.
P_LABOR_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged labor transactions. This is defaulted to a project from the project template. If cc_process_labor_flag is set to 'Y', this field is required.
P_LABOR_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for labor transactions. This is defaulted to a project from the project template. This value for the project is default for the task fixed date. If cc_process_labor flag is set to 'Y', this field is required.
P_CC_PROCESS_NL_FLAG	IN	VARCHAR2(1)	No	Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Defaulted value for the project template is 'N'. This is defaulted to a project from the project template.

Name	Usage	Type	Req?	Description
P_NL_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If cc_process_nl_labor flag is set to 'Y', this field is required.
P_NL_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for non-labor transactions. This is defaulted to a project from the project template. If cc_process_nl_flag is set to'Y', this field is required.
P_CC_TAX_TASK_ID	IN	NUMBER	No	Identifier of the task to which intercompany tax items on the intercompany AP invoice are charged.
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfields
P_ROLE_LIST_ID	IN	NUMBER(15)	No	Identifier of the role list, a list of allowable roles that are displayed when team members are assigned
P_WORK_TYPE_ID	IN	NUMBER(15)	No	Work type identifier. Work types are predefined types of work. For example, Vacation, Training, and Administration.
P_CALENDAR_ID	IN	NUMBER(15)	No	Calendar identifier. A calendar specifies exceptions such as public holidays.
P_LOCATION_ID	IN	NUMBER(15)	No	Identifier of the project work site location
P_PROBABILITY_MEMBER_ID	IN	NUMBER(15)	No	Identifier of the probability member. Project probability, the likelihood that a project will be approved, is used as a weighting average for reporting.
P_PROJECT_VALUE	IN	NUMBER	No	The opportunity value converted to the project functional currency

Name	Usage	Type	Req?	Description
P_EXPECTED_APPROVAL_DATE	IN	DATE	No	The expected date of the project approval (for information purposes only)
P_INITIAL_TEAM_TEMPLATE_ID	IN	NUMBER(15)	No	The team template that you want to add to a new project
P_JOB_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	The identifier of the job-based bill rate schedule for the project
P_EMP_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	The identifier of the employee-based bill tate schedule for the project
P_COMPETENCE_MATCH_WT	IN	NUMBER	No	The weighting value for competence match, used to calculate the score
P_AVAILABILITY_MATCH_WT	IN	NUMBER	No	The weighting value for availability match, used to calculate the score
P_JOB_LEVEL_MATCH_WT	IN	NUMBER	No	The weighting value for job-level match, used to calculate the score
P_ENABLE_AUTOMATED_SEARCH	IN	VARCHAR2(1)	No	Flag that indicates whether automated candidate nomination is used for the requirements on a project
P_SEARCH_MIN_AVAILABILITY	IN	NUMBER	No	The minimum required availability for a resource to be returned in the search result
P_SEARCH_ORG_HIER_ID	IN	NUMBER(15)	No	Organization hierarchy for searches
P_SEARCH_STARTING_ORG_ID	IN	NUMBER(15)	No	Starting organization for searches
P_SEARCH_COUNTRY_CODE	IN	VARCHAR2(2)	No	Country for searches
P_MIN_CAND_SCORE_REQD_FOR_NOM	IN	NUMBER	No	Minimum score required for a resource to be nominated as candidate on a requirement
P_NON_LAB_STD_BILL_RT_SCH_ID	IN	NUMBER(15)	No	Identifier of the non-labor standard bill rate schedule
P_INVPROC_CURRENCY_TYPE	IN	VARCHAR2(30)	No	Invoice processing currency code
P_REVPROC_CURRENCY_CODE	IN	VARCHAR2(15)	No	Revenue processing currency code in the project functional currency
P_PROJECT_BIL_RATE_DATE_CODE	IN	VARCHAR2(30)	No	Exchange rate date type for converting customer billing amounts from bill transaction currency or funding currency to project currency

Name	Usage	Type	Req?	Description
P_PROJECT_BIL_RATE_TYPE	IN	VARCHAR2(30)	No	Exchange rate type for converting customer billing amounts from bill transaction currency or funding currency to project currency
P_PROJECT_BIL_RATE_DATE	IN	DATE	No	Exchange rate date for converting customer billing amounts from bill transaction currency or funding currency to project currency, if the rate date type is Fixed.
P_PROJECT_BIL_EXCHANGE_RATE	IN	NUMBER	No	Exchange rate for conversion from bill transaction currency or funding currency to project currency if the rate type is User
P_PROJFUNC_CURRENCY_CODE	IN	VARCHAR2(15)	No	Project functional currency. The default value is the value entered for the associated set of books.
P_PROJFUNC_BIL_RATE_DATE_CODE	IN	VARCHAR2(30)	No	Exchange rate date type for converting customer billing amounts from bill transaction currency or funding currency to project functional currency
P_PROJFUNC_BIL_RATE_TYPE	IN	VARCHAR2(30)	No	Exchange rate type for converting customer billing amounts from bill transaction currency or funding currency to project functional currency
P_PROJFUNC_BIL_RATE_DATE	IN	DATE	No	Exchange rate date for converting customer billing amounts from bill transaction currency or funding currency to project functional currency if the rate date type is Fixed
P_PROJFUNC_BIL_EXCHANGE_RATE	IN	NUMBER	No	Exchange rate for conversion from bill transaction currency or funding currency to project functional currency if the rate type is User
P_FUNDING_RATE_DATE_CODE	IN	VARCHAR2(30)	No	Exchange rate date type for conversting customer billing amounts from bill transaction currency to funding currency

Name	Usage	Type	Req?	Description
P_FUNDING_RATE_TYPE	IN	VARCHAR2(30)	No	Exchange rate type for converting customer billing amounts from bill transaction currency to funding currency
P_FUNDING_RATE_DATE	IN	DATE	No	Exchange rate date for converting customer billing amounts from bill transaction currency to funding currency if rate date type is Fixed
P_FUNDING_EXCHANGE_RATE	IN	NUMBER	No	Exchange rate for conversion from bill transaction currency to project or functional currency if rate type is User
P_BASELINE_FUNDING_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether the funding can be baselined without a revenue budget
P_PROJFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	Default value for the project functional cost rate
P_PROJFUNC_COST_RATE_DATE	IN	DATE	No	Default value for the project functional cost rate date
P_INV_BY_BILL_TRANS_CURR_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether invoicing is by bill transaction currency for the project
P_MULTI_CURRENCY_BILLING_FLAG	IN	VARCHAR2(1)	No	Flag that indicates if multi-currency billing is allowed for the project
P_ASSIGN_PRECEDES_TASK	IN	VARCHAR2(1)	No	Flag that indicatesd if assignment level attributes override task level attributes
P_PRIORITY_CODE	IN	VARCHAR2(30)	No	The code identifying the priority of the project
P_RETN_BILLING_INV_FORMAT_ID	IN	NUMBER(15)	No	The identifier of the retention billing invoice format
P_RETN_ACCOUNTING_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether retention accounting is enabled for the project
P_ADV_ACTION_SET_ID	IN	NUMBER(15)	No	Flag that indicates the default advertisement action set of the project or project template
P_START_ADV_ACTION_SET_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether the advertisement action set will start immediately after a requirement is created

Name	Usage	Type	Req?	Description
P_REVALUATE_FUNDING_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether the funding has to be revaluated
P_INCLUDE_GAINS_LOSSES_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether gains and losses to be included in project revenue
P_TARGET_START_DATE	IN	DATE	No	The target start date for the project
P_TARGET_FINISH_DATE	IN	DATE	No	The target finish date for the project
P_BASELINE_START_DATE	IN	DATE	No	The baseline start date of the project
P_SELINE_FINISH_DATE	IN	DATE	No	The baseline finish date of the project
P_SCHEDULED_AS_OF_DATE	IN	DATE	No	The publish date for the scheduled start and finish dates for the project
P_BASELINE_AS_OF_DATE	IN	DATE	No	The baseline date for the baseline start and finish dates for the project
P_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	Reason code for labor discount
P_NON_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	Reason code for non-labor discount
P_SECURITY_LEVEL	IN	NUMBER	No	Indicates whether a project is public or private. Zero (0) indicates that the project is private. 100 indicates that the project is public.
P_ACTUAL_AS_OF_DATE	IN	DATE	No	The publish date for the project actual start and actual finish dates
P_SCHEDULED_DURATION	IN	NUMBER	No	Duration from the scheduled start date to the scheduled finish date using the project work calendar
P_BASELINE_DURATION	IN	NUMBER	No	Duration from the baseline start date to the baseline finish date using the project work calendar
P_ACTUAL_DURATION	IN	NUMBER	No	Duration from the actual start date to the actual finish date using the project work calendar
P_LONG_NAME	IN	VARCHAR2(240)	No	Project long name
P_BTC_COST_BASE_REV_CODE	IN	VARCHAR2(90)	No	Bill transaction currency for cost-based revenue

Name	Usage	Type	Req?	Description
P_ASSET_ALLOCATION_METHOD	IN	VARCHAR2(30)	No	The method used to allocate indirect and common costs across the assets assigned to a grouping level
P_CAPITAL_EVENT_PROCESSING	IN	VARCHAR2(30)	No	The capital event processing method, used to determine when cost and assets are grouped for capitalization or retirement adjustment processing
P_CINT_RATE_SCH_ID	IN	NUMBER(15)	No	Identifier of the capital interest rate schedule
P_CINT_ELIGIBLE_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether the project is eligible for capitalized interest
P_CINT_STOP_DATE	IN	DATE	No	Stop date for capital interest calculation

LOAD_TASK

LOAD_TASK is a Load–Execute–Fetch procedure used to load a task to a global PL / SQL table.

Business Rule (task level)

Oracle Projects imposes the following business rule:

- Parent tasks must be loaded before their subtasks.

The following table shows the parameters for LOAD_TASK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that identifies a project's task in the external system
P_PA_TASK_ID	IN	NUMBER	No	For descriptions of this and the following parameters, see: TASK_IN_TBL_TYPE: page 3 – 15
P_TASK_NAME	IN	VARCHAR2(20)	Yes	
P_PA_TASK_NUMBER	IN	VARCHAR2(25)	Yes	
P_TASK_DESCRIPTION	IN	VARCHAR2(250)	No	
P_TASK_START_DATE	IN	DATE	No	
P_TASK_COMPLETION_DATE	IN	DATE	No	
P_PM_PARENT_TASK_REFERENCE	IN	VARCHAR2(25)	No	
P_PA_PARENT_TASK_ID	IN	NUMBER	No	
P_ADDRESS_ID	IN	NUMBER	No	
P_CARRYING_OUT_ORGANIZATION_ID	IN	NUMBER(15)	No	
P_SERVICE_TYPE_CODE	IN	VARCHAR2(30)	No	
P_TASK_MANAGER_PERSON_ID	IN	NUMBER(9)	No	
P_BILLABLE_FLAG	IN	VARCHAR2(1)	No	
P_CHARGEABLE_FLAG	IN	VARCHAR2(1)	No	
P_READY_TO_BILL_FLAG	IN	VARCHAR2(1)	No	
P_READY_TO_DISTRIBUTE_FLAG	IN	VARCHAR2(1)	No	
P_LIMIT_TO_TXN_CONTROLS_FLAG	IN	VARCHAR2(1)	No	
P_LABOR_BILL_RATE_ORG_ID	IN	NUMBER(15)	No	

Name	Usage	Type	Req?	Description
P_LABOR_STD_BILL_RATE_SCHDL	IN	VARCHAR2(20)	No	
P_LABOR_SCHEDULE_FIXED_DATE	IN	DATE	No	
P_LABOR_SCHEDULE_DISCOUNT	IN	NUMBER(7,4)	No	
P_NL_BILL_RATE_ORG_ID	IN	NUMBER(15)	No	
P_NL_STD_BILL_RATE_SCHDL	IN	VARCHAR2(30)	No	
P_NL_SCHEDULE_FIXED_DATE	IN	DATE	No	
P_NL_SCHEDULE_DISCOUNT	IN	NUMBER(7,4)	No	
P_LABOR_COST_MULTIPLIER_NAME	IN	VARCHAR2(20)	No	
P_COST_IND_RATE_SCH_ID	IN	NUMBER(15)	No	
P_REV_IND_RATE_SCH_ID	IN	NUMBER(15)	No	
P_INV_IND_RATE_SCH_ID	IN	NUMBER(15)	No	
P_COST_IND_SCH_FIXED_DATE	IN	DATE	No	
P_REV_IND_SCH_FIXED_DATE	IN	DATE	No	
P_INV_IND_SCH_FIXED_DATE	IN	DATE	No	
P_LABOR_SCH_TYPE	IN	VARCHAR2(1)	No	
P_NL_SCH_TYPE	IN	VARCHAR2(1)	No	
P_ACTUAL_START_DATE	IN	DATE	No	
P_ACTUAL_FINISH_DATE	IN	DATE	No	
P_EARLY_START_DATE	IN	DATE	No	
P_EARLY_FINISH_DATE	IN	DATE	No	
P_LATE_START_DATE	IN	DATE	No	
P_LATE_FINISH_DATE	IN	DATE	No	
P_SCHEDULED_START_DATE	IN	DATE	No	
P_SCHEDULED_FINISH_DATE	IN	DATE	No	
P_ALLOW_CROSS_CHARGE_FLAG	IN	VARCHAR2(1)	No	Cross charge allowed? Value is required. Default Value is 'N'. This value can be overridden at any task level.
P_PROJECT_RATE_DATE	IN	DATE	No	Default project currency rate date (date for accounting currency rate for a given rate type).
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Default project currency rate type (e.g., Spot, Corporate).

Name	Usage	Type	Req?	Description
P_CC_PROCESS_LABOR_FLAG	IN	VARCHAR2(1)	No	Flag that indicates cross charge processing is to be performed for labor transactions charged to the project. Default value for the project template is 'N'. This is defaulted to a project from the project template.
P_LABOR_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged labor transactions. This is defaulted to a project from the project template. If cc_process_labor_flag is set to 'Y', this field is required.
P_LABOR_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for labor transactions. This is defaulted to a project from the project template. This value for the project is default for the task fixed date. If cc_process_labor flag is set to 'Y', this field is required.
P_CC_PROCESS_NL_FLAG	IN	VARCHAR2(1)	No	Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Defaulted value for the project template is 'N'. This is defaulted to a project from the project template.
P_NL_TP_SCHEDULE_ID	IN	NUMBER	No	Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If cc_process_nl_labor flag is set to 'Y', this field is required.

Name	Usage	Type	Req?	Description
P_NL_TP_FIXED_DATE	IN	DATE	No	Fixed date to find the effective rate of the bill rate or burden schedule when determining the transfer price for non-labor transactions. This is defaulted to a project from the project template. If cc_process_nl_flag is set to'Y', this field is required.
P_RECEIVE_PROJECT_INVOICE_FLAG	IN	VARCHAR2(1)	No	Flag that indicates that the task may recive charges from internal suppliers via inter-project billing.
P_NL_SCH_TYPE	IN	VARCHAR2(1)	No	
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	
P_OBLIGATION_START_DATE	IN	PA_DATE_1000_DATE	No	The start date of the obligation
P_OBLIGATION_FINISH_DATE	IN	PA_DATE_1000_DATE	No	The finish date of the obligation
P_BASELINE_START_DATE	IN	PA_DATE_1000_DATE	No	The baseline start date
P_BASELINE_FINISH_DATE	IN	PA_DATE_1000_DATE	No	The baseline finish date
P_CLOSED_DATE	IN	PA_DATE_1000_DATE	No	The closed date
P_WQ_UOM_CODE	IN	PA_VC_1000_150	No	The unit of measure used for work quantity for the task
P_WQ_ITEM_CODE	IN	PA_VC_1000_150	No	The work item code for work quantity for the task
P_STATUS_CODE	IN	PA_VC_1000_150	No	Status of the task
P_WF_STATUS_CODE	IN	PA_VC_1000_150	No	The workflow status code
P_PM_SOURCE_CODE	IN	PA_VC_1000_150	No	The product code of the project management tool supplier
P_CALENDAR_ID	IN	PA_NUM_1000_NUM	No	The identier of the calendar associated with the task
P_PLANNED_EFFORT	IN	PA_NUM_1000_NUM	No	The planned effort for the task
P_PLANNED_WORK_QUANTITY	IN	PA_NUM_1000_NUM	No	The planned work quantity
P_TASK_TYPE	IN	PA_NUM_1000_NUM	No	The unique identifier for the task type
P_JOB_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	The identifier of the job-based bill rate schedule for the project

Name	Usage	Type	Req?	Description
P_EMP_BILL_RATE_SCHEDULE_ID	IN	NUMBER	No	The identifier of the employee-based bill rate schedule for the project
P_TASKFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The task-level default value for project functional cost rate type
P_TASKFUNC_COST_RATE_DATE	IN	DATE	No	The task-level default value for project functional cost rate date
P_NON_LAB_STD_BILL_RT_SCH_ID	IN	NUMBER(15)	No	Identifier of the non-labor standard bill rate schedule
P_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	Reason code for labor discount
P_NON_LABOR_DISC_REASON_CODE	IN	VARCHAR2(30)	No	Reason code for non-labor discount
P_LONG_TASK_NAME	IN	VARCHAR2(240)	No	Task long name
P_RETIREMENT_COST_FLAG	IN	VARCHAR2(1)	No	Flag that identifies tasks for retirement cost collection
P_CINT_ELIGIBLE_FLAG	IN	VARCHAR2(1)	No	Flag that indicates whether the project is eligible for capitalized interest
P_CINT_STOP_DATE	IN	DATE	No	Stop date for capital interest calculation
P_REVENUE_ACCRUAL_METHOD	IN	VARCHAR2(30)	No	The revenue accrual method for task
P_INVOICE_METHOD	IN	VARCHAR2(30)	No	The invoice method for the task
P_OBLIGATION_START_DATE	IN	DATE	No	The obligation start date of the workplan version
P_OBLIGATION_FINISH_DATE	IN	DATE	No	The obligation finish date of the workplan version
P_ACTUAL_START_DATE	IN	DATE	No	The actual start date of the workplan version
P_ACTUAL_FINISH_DATE	IN	DATE	No	The actual end date of the workplan version
P_ESTIMATED_START_DATE	IN	DATE	No	The estimated start date of the workplan version
P_ESTIMATED_FINISH_DATE	IN	DATE	No	The estimated finish date of the workplan version
P_EARLY_START_DATE	IN	DATE	No	The early start date of the workplan version
P_EARLY_FINISH_DATE	IN	DATE	No	The early finish date of the workplan version
P_LATE_START_DATE	IN	DATE	No	The late start date of the workplan version
P_LATE_FINISH_DATE	IN	DATE	No	The late finish date of the workplan version

Name	Usage	Type	Req?	Description
P_MILESTONE_FLAG	IN	VARCHAR2(1)	No	Flag that indicates if the task version is a milestone. This is a task-specific attribute.
P_CRITICAL_FLAG	IN	VARCHAR2(1)	No	Flag that indicates if the task version is part of the critical path. This is a task-specific attribute.
P_WQ_PLANNED_QUANTITY	IN	NUMBER(17)	No	The planned work quantity for the task
P_PLANNED_EFFORT	IN	NUMBER(17)	No	The planned effort for the task

LOAD_TASKS

LOAD_TASKS is a Load–Execute–Fetch procedure used to load tasks to a global PL/SQL table. The parameters for this procedure are the same as the parameters for LOAD_TASK: page 3 – 55.

Check Procedures

The following check procedures are PL/SQL procedures used to verify in real time that:

- Project and task information you have entered into your external system is unique in Oracle Projects
- Certain functions, such as deleting a project or task, follow the business rules defined in Oracle Projects

CHECK_ADD_SUBTASK_OK

Use the Check procedure CHECK_ADD_SUBTASK_OK to determine if a subtask can be added to a parent task.

The following table shows the parameters for CHECK_ADD_SUBTASK_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that identifies the task in the external system

Name	Usage	Type	Req?	Description
P_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_ADD_SUBTASK_OK_FLAG	OUT	VARCHAR2(1)		Indicates whether or not a subtask can be added to this task (Y or N)

CHECK_CHANGE_PARENT_OK

Use the Check procedure CHECK_CHANGE_PARENT_OK to determine if you can move a task from one parent task to another. You can move a task as long as it retains the same top task.

The following table shows the parameters for CHECK_CHANGE_PARENT_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the task in the external system
P_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_NEW_PARENT_TASK_ID	IN	NUMBER(15)		The Oracle Projects identification code of the new parent task
P_PM_NEW_PARENT_TASK_REFERENCE	IN	VARCHAR2(25)		The external system reference code of the new parent task
P_CHANGE_PARENT_OK_FLAG	OUT	VARCHAR2(1)		Indicates whether or not this task can be assigned to a new parent task (Y or N)

CHECK_CHANGE_PROJECT_ORG_OK

Use the Check procedure CHECK_CHANGE_PROJECT_ORG_OK to determine if you can change the CARRYING_OUT_ORGANIZATION_ID field for a particular project or task.

The following table shows the parameters for CHECK_CHANGE_PROJECT_ORG_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_CHANGE_PROJECT_ORG_OK_FLAG	OUT	VARCHAR2(1)		Indicates whether or not the carrying out organization of this project can be changed (Y or N)

CHECK_DELETE_PROJECT_OK

Use the Check procedure CHECK_DELETE_PROJECT_OK to determine if you can delete a project.

The following table shows the parameters for CHECK_DELETE_PROJECT_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

Name	Usage	Type	Req?	Description
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_DELETE_PROJECT_OK_FLAG	OUT	VARCHAR2(1)		Indicates whether or not this project may be deleted (Y or N)

CHECK_DELETE_TASK_OK

Use the Check procedure CHECK_DELETE_TASK_OK to determine if you can delete a task.

The following table shows the parameters for CHECK_DELETE_TASK_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the task in the external system
P_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_DELETE_TASK_OK_FLAG	OUT	VARCHAR2(1)		Indicates whether or not this task can be deleted (Y or N)

CHECK_TASK_NUMBER_CHANGE_OK

Use the Check procedure CHECK_TASK_NUMBER_CHANGE_OK to determine if you can change a task's number.

The following table shows the parameters for CHECK_TASK_NUMBER_CHANGE_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	BOOLEAN		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the task in the external system
P_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies a task within a project in Oracle Projects
P_TASK_NUMBER_CHANGE_OK_FLAG	OUT	VARCHAR2(1)		Indicates whether or not the task number can be changed (Y or N)

CHECK_UNIQUE_PROJECT_REFERENCE

Use the Check procedure CHECK_UNIQUE_PROJECT_REFERENCE to determine if a new or changed project reference (PM_PROJECT_REFERENCE) is unique.

The following table shows the parameters for CHECK_UNIQUE_PROJECT_REFERENCE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	BOOLEAN		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard

Name	Usage	Type	Req?	Description
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_UNIQUE_PROJECT_REF_FLAG	OUT	VARCHAR2(1)		Indicates whether or not this project reference is unique in Oracle Projects (Y or N)

CHECK_UNIQUE_TASK_NUMBER

Use the Check procedure CHECK_UNIQUE_TASK_NUMBER to determine if a new or changed task number is unique within a project.

The following table shows the parameters for CHECK_UNIQUE_TASK_NUMBER.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_TASK_NUMBER	IN	VARCHAR2(25)	No	The number that identifies the task in Oracle Projects
P_UNIQUE_TASK_NUMBER_FLAG	OUT	VARCHAR2(1)		Indicates whether or not this task number is unique in the project within Oracle Projects (Y or N)

CHECK_UNIQUE_TASK_REFERENCE

Use the Check procedure CHECK_UNIQUE_TASK_REFERENCE to determine if a new or changed task reference (PM_TASK_REFERENCE) is unique.

The following table shows the parameters for CHECK_UNIQUE_TASK_REFERENCE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)		API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system
P_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the task in the external system
P_UNIQUE_TASK_REF_FLAG	OUT	VARCHAR2(1)		Shows if this task reference is unique in this project within Oracle Projects (Y or N)

The following example describes how to create an interface between Oracle Projects and the project and task information entered in your system. Depending on your company's business needs, your implementation of the project APIs may be more or less complex than the scenario shown here. As you work through the example, you may want to refer to information elsewhere in the manual:

- For a detailed description of the project APIs, see Project APIs: page 3 – 2.
- Most of the Oracle Projects APIs use a standard set of input and output parameters. For a description of these parameters, see Standard API Parameters: page 2 – 19.
- For an example of PL / SQL code for creating a project without using composite datatypes, see Creating a Project Using the Load–Execute–Fetch APIs: page 3 – 76.

Step 1 Connect to an Oracle database

To ensure that proper security is enforced while accessing Oracle Projects data, follow the steps in Security Requirements: page 2 – 9.

Step 2 Select a source template or project

When using the APIs to create a new project in Oracle Projects, first select a project template from which to create the new project. Oracle Projects will not create a new project unless you perform this step. Use the API view PA_SELECT_TEMPLATE_V to select a valid Oracle Projects source template.

Alternatively, you can choose a source project. The only difference between templates and projects is that the field TEMPLATE_FLAG for templates is set to Y. All projects originate from templates, and the originating template determines which Quick Entry fields appear in your new project. In this section, all instructions involving source templates also apply to source projects.

Step 3 Get the Quick Entry fields of the source template

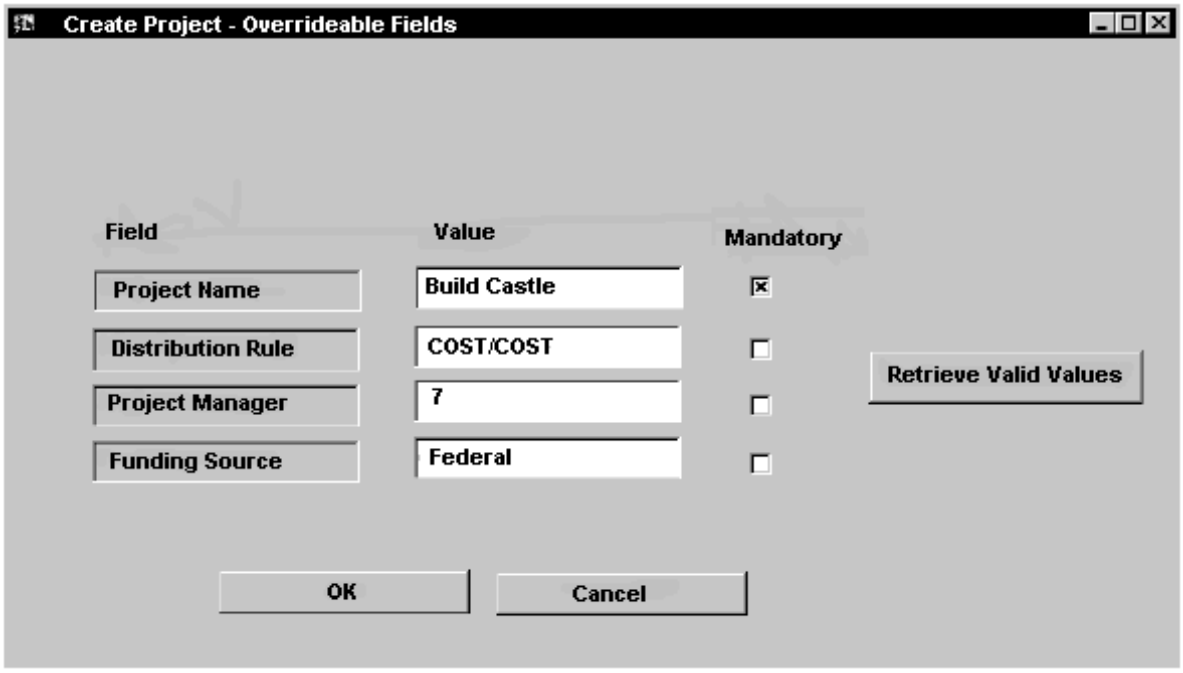
After you select a source template, use the PA_SOURCE_TEMPLATE_ID to retrieve the Quick Entry fields associated with the template. You assign Quick Entry fields to a template when you create the template in Oracle Projects. For more information about Quick Entry fields, see

PA_PROJECT_COPY_OVERRIDES in Oracle eTRM, available on Oracle*Metalink*.

The view PA_OVERRIDE_FIELDS_V displays all the Quick Entry fields associated with a particular template. The user interface you design should display at least the Display Name, Value, and Mandatory fields and should allow users to enter information only into the Value field.

An example of a user interface that meets these requirements is shown below:

Figure 3 – 1Example of a Quick Entry (Overridable Fields) Window



Step 4 Enter valid data for the Quick Entry fields

Lists of values (LOVs) validate most of the Quick Entry fields. The view PA_OVERRIDE_FIELDS_V retrieves the name of the view that contains the valid data for the active row and returns this name in the field LOV_VIEW_NAME. Your project management tool can use this information to dynamically access the appropriate view.

For example, if you place your cursor in the Funding Source field and choose Retrieve Valid Values, your project management tool will display a screen with two columns, Code and Description. Values

retrieved from the database view PA_CLASS_CATEGORIES_LOV_V will appear under these two column headings.

You can also use the following views to retrieve lists of values for a project's Quick Entry fields:

- PA_PROJECT_STATUS_CODES_LOV_V
- PA_DISTRIBUTION_RULES_LOV_V
- PA_KEY_MEMBERS_LOV_V
- PA_ORGANIZATIONS_LOV_V
- PA_CUSTOMERS_LOV_V

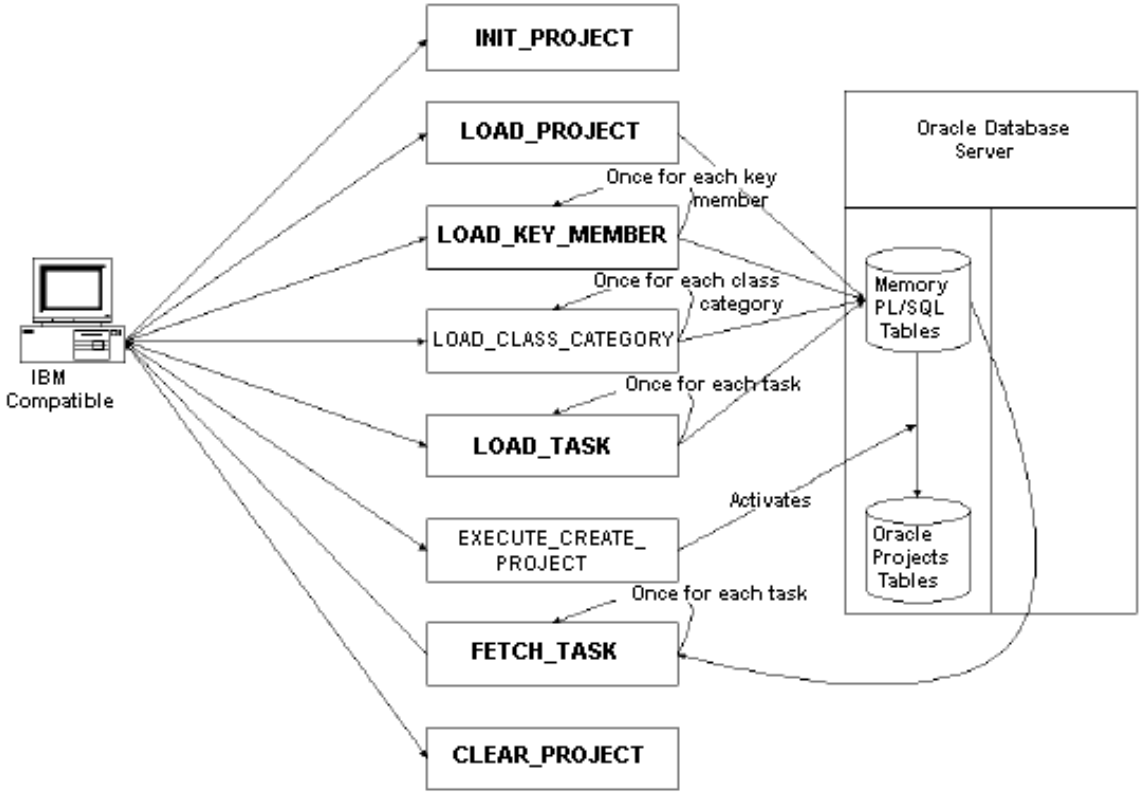
Step 5 Interface project information to the server

Not all tools can call the APIs that use composite datatypes. Tools that do not support composite datatypes must call the supplementary Load–Execute–Fetch APIs. The Load–Execute–Fetch APIs include procedures to initialize, load, execute, fetch, and clear data.

Use these APIs only if you use a tool that does not support composite datatype parameters. If the tool (for example, Oracle PL/SQL Version 2.3 or higher) supports composite datatype parameters, you can call the CREATE_PROJECT and UPDATE_PROJECT APIs directly.

The following illustration shows the flow of the Load–Execute–Fetch project procedures.

Figure 3 – 2 **Load–Execute–Fetch Procedures for Project APIs**



In the example above, INIT_PROJECT resets the server–side global PL / SQL tables that temporarily store the project and task data.

Once you set up these tables, you can use LOAD_PROJECT to move the project data to the Oracle Projects database.

The following table illustrates the relationship between the information in the user interface and LOAD_PROJECT:

Quick Entry Field Value	*LOAD_PROJECT Parameter*
NAME	P_PROJECT_NAME
DESCRIPTION	P_DESCRIPTION
START_DATE	P_START_DATE
COMPLETION_DATE	P_COMPLETION_DATE
PROJECT_STATUS_CODE	P_PROJECT_STATUS_CODE
PUBLIC_SECTOR_FLAG	P_PUBLIC_SECTOR_FLAG
DISTRIBUTION_RULE	P_DISTRIBUTION_RULE
CARRYING_OUT_ ORGANIZATION_ID	P_CARRYING_OUT_ ORGANIZATION_ID
CUSTOMER_NAME	P_CUSTOMER_ID

Table 3 – 5 How the User Interface Relates to LOAD_PROJECT (Page 1 of 1)

LOAD_PROJECT passes the values entered into the Quick Entry value field to their corresponding parameters. LOAD_PROJECT passes additional parameters, depending on whether you are updating an existing project or creating a new one. When you create a new project, this procedure must also pass the following parameters:

- P_PM_PROJECT_REFERENCE passes the unique reference code that identifies the project in the external system.
- P_CREATED_FROM_PROJECT_ID passes the unique reference code that identifies the source template in Oracle Projects (PA_SOURCE_TEMPLATE_ID).

If your project has multiple key members or class categories, you must call the APIs LOAD_KEY_MEMBER and LOAD_CLASS_CATEGORY for every key member and class category associated with your project.

During project creation, the Quick Entry fields Key Members and Class Category are related to the input parameters shown in the two tables that follow.

The following table shows the input parameters for the key member quick entry field.

Key Member Quick Entry Field	*Input Parameter for LOAD_KEY_MEMBERS*
KEY_MEMBER (value)	P_PERSON_ID
KEY_MEMBER (display_name)	P_PROJECT_ROLE_TYPE

Table 3 – 6 Key Member Quick Entry Field (Page 1 of 1)

The following table shows the input parameters for the class category quick entry field.

Class Category Quick Entry Field	*Input Parameter for LOAD_CLASS_CATEGORY*
CLASS_CATEGORY (value)	P_CLASS_CODE
CLASS_CATEGORY (display_name)	P_CLASS_CATEGORY

Table 3 – 7 Class Category Quick Entry Field (Page 1 of 1)

Step 6 Interface task information to the server

After you interface the project–related data to the server, you can call LOAD_TASK to interface task–related data to the server–side global PL/SQL tables. Call LOAD_TASK once for every task in the project.

Attention: You must load parent tasks before you can load their subtasks.

Each task must specify at least the following information:

- P_PM_TASK_REFERENCE**.** The unique reference code that identifies the task in the external system.
- P_PM_PARENT_TASK_REFERENCE**.** The unique reference code that identifies the task's parent task. This parameter is left blank for top tasks.
- P_TASK_NAME**.** The name of the task.

For the names and descriptions of other parameters that LOAD_TASK can pass, see LOAD_TASK: page 3 – 55.

Step 7 Start the server–side process

Once the Load procedures have successfully moved project and task data to the Oracle Projects global PL/SQL tables, call the procedure EXECUTE_CREATE_PROJECT to process the project and task data that you interfaced to the global PL/SQL tables. In addition to the standard input and output parameters, this Execute procedure requires the following parameters:

- Input parameter: P_PM_PRODUCT_CODE, the identification code of the product exporting the project. For information about setting up your product (external system) as a source, refer to Setting Up Your Product in Oracle Projects: page 2 – 8.
- Output parameters:
  - P_PA_PROJECT_ID, the unique Oracle Projects identification code for the new project.
  - P_PA_PROJECT_NUMBER, the unique Oracle Projects number for the new project. If you have set up Oracle Projects to support manual project numbering, P_PA_PROJECT_NUMBER should be identical to the P_PM_PROJECT_REFERENCE. If you have implemented automatic numbering, this parameter returns an automatically generated number.

Step 8 Get return values for tasks

After the Load and Execute procedures create your project and tasks in Oracle Projects, use FETCH_TASK to return each unique task identification code from Oracle Projects. The key parameters for this procedure are the input parameter P_TASK_INDEX, which points to a single task, and the output parameters P_PA_TASK_ID and P_PM_TASK_REFERENCE.

To call the procedure for each task, you can write a simple program to call FETCH_TASK in a loop with P_TASK_INDEX as the stepping variable (1 through the total number of tasks). The output parameter P_TASK_RETURN_STATUS indicates whether the API handled the specific task successfully ('S'). If the parameter returns an 'E' or 'U', the task caused an error, and you must stop the Fetch procedure to retrieve the related error message. Fetch APIs do not return error message data. Instead, use GET_MESSAGES to retrieve the error text, as described in the next step.

Step 9 Retrieve error messages

Every Oracle Projects API includes two standard output parameters: P_RETURN_STATUS indicates whether the API was executed successfully, and P_MSG_COUNT shows the number of errors detected during the execution of the API.

If the API detects one error, the API returns the error message text. If the API detects multiple errors, use GET_MESSAGES to retrieve the error messages. See GET_MESSAGES: page 2 – 24.

Step 10 Finish the Load–Execute–Fetch process

After executing the Fetch procedures and retrieving any error messages, finish the Load–Execute–Fetch process by calling the API CLEAR_PROJECT and either save or rollback your changes to the database.

Creating a Project Using the Load–Execute–Fetch APIs

The following PL/SQL code is a sample of a script that you can use to create a project using the Load–Execute–Fetch APIs.

The Load–Execute–Fetch APIs use parameters with standard datatypes (VARCHAR2, NUMBER, and DATE). They do not use composite datatypes.

```
DECLARE
--variables needed to create task hierarchy

level1                          NUMBER;
level2                          NUMBER;
level3                          NUMBER;
a                                       NUMBER := 0;
m                                       NUMBER := 0;
parent_level1                   VARCHAR2(30);
parent_level2                   VARCHAR2(30);
parent_level3                   VARCHAR2(30);
number_of_tasks1                NUMBER; --number of tasks/level
number_of_tasks2                NUMBER;
number_of_tasks3                NUMBER;
number_of_tasks4                NUMBER;

--variables needed for API standard parameters

l_api_version_number            NUMBER  :=1.0           ;
l_commit                        VARCHAR2(1):= 'F';
l_return_status                 VARCHAR2(1);
l_init_msg_list                 VARCHAR2(1);
l_msg_count             NUMBER;
l_msg_data                      VARCHAR2(2000);
l_data                          VARCHAR2(2000);
l_msg_entity                    VARCHAR2(100);
l_msg_entity_index      NUMBER;
l_msg_index             NUMBER;
l_msg_index_out                 NUMBER;
l_encoded                       VARCHAR2(1);

--variables needed for Oracle Project specific parameters

l_created_from_project_id NUMBER;
l_pm_product_code       VARCHAR2(10);
l_number_of_task_levels         NUMBER;
l_project_name                  VARCHAR2(30);
l_pm_project_reference          VARCHAR2(25);
l_project_status_code           VARCHAR2(30);
l_distribution_rule             VARCHAR2(30);
```

```
l_public_sector_flag              VARCHAR2(1);
l_carrying_out_organization_id    NUMBER;
l_start_date                      DATE;
l_completion_date         DATE;
l_actual_start_date               DATE;
l_actual_finish_date              DATE;
l_early_start_date        DATE;
l_early_finish_date               DATE;
l_late_start_date         DATE;
l_late_finish_date        DATE;
l_person_id               NUMBER;
l_project_role_type               VARCHAR2(20);
l_class_category                  VARCHAR2(30);
l_class_code                      VARCHAR2(30);
l_project_id                      NUMBER(15);
l_pa_project_number               VARCHAR2(25);
l_project_description             VARCHAR2(250);
l_customer_id                     NUMBER;
l_project_relationship_code       VARCHAR2(30);
l_task_id                         NUMBER(15);
l_pm_task_reference               VARCHAR2(25);
l_task_index                      NUMBER;
l_tasks_in                        pa_project_pub.task_in_tbl_type;
l_task_rec                        pa_project_pub.task_in_rec_type;
l_key_member_rec                  pa_project_pub.project_role_rec_t
ype;
l_key_member_tbl                  pa_project_pub.project_role_tbl_t
ype;
l_task_return_status              VARCHAR2(1);
API_ERROR                         EXCEPTION;

BEGIN

--PRODUCT RELATED DATA
l_pm_product_code :='SOMETHING';

--PROJECT DATA
l_created_from_project_id := 1040;
l_project_name := 'PROJECT_NAME';
l_pm_project_reference := 'PROJECT_NAME';
l_project_description := 'PROJECT_DESCRIPTION';
l_project_status_code := '';
--l_distribution_rule := 'COST/COST';
l_carrying_out_organization_id :=2;
l_start_date :='01-jan-94';
l_completion_date :='31-mar-99';
l_actual_start_date :='01-jan-93';
l_actual_finish_date :='01-apr-99';
l_early_start_date :='01-jan-94';
```

```
l_early_finish_date :='31-mar-99';
l_late_start_date :='01-jan-94';
l_late_finish_date :='31-mar-99';

--KEY MEMBERS DATA

m:= 1;

l_person_id :='29';
l_project_role_type :='PROJECT MANAGER';
l_key_member_rec.person_id :=29;
l_key_member_rec.project_role_type :='PROJECT MANAGER';
l_key_member_tbl(m) := l_key_member_rec;

m:=2;

l_key_member_rec.person_id:=30;
l_key_member_rec.project_role_type :='Project Coordinator';

l_key_member_tbl(m) := l_key_member_rec;

m:=3;

l_key_member_rec.person_id :=7;
l_key_member_rec.project_role_type :='Project Coordinator';

l_key_member_tbl(m) := l_key_member_rec;

--CLASS CATEGORIES DATA
l_class_category :='Funding Source';
l_class_code :='Federal';


--TASKS DATA

--Set the number of tasks for every level (there are 4 levels)
number_of_tasks1 := 10;
number_of_tasks2 := 1;
number_of_tasks3 := 1;
number_of_tasks4 := 0;

for level1 in 1..number_of_tasks1 loop

    a:= a + 1;
    l_task_rec.pm_task_reference :=a;
    l_task_rec.task_name :='TOP LEVEL '||a;
    l_task_rec.pm_parent_task_reference :='';
      l_task_rec.task_start_date := '09-MAR-95';
      l_task_rec.task_completion_date := '05-JUL-95';
```

```
      l_task_rec.actual_start_date := '10-MAR-95';
      l_task_rec.actual_finish_date := '06-JUL-95';
      l_task_rec.early_start_date := '09-MAR-95';
      l_task_rec.early_finish_date := '05-JUL-95';
      l_task_rec.late_start_date := '09-MAR-95';
      l_task_rec.late_finish_date := '05-JUL-95';

    --l_task_rec.address_id := 1012;
    l_tasks_in(a):= l_task_rec;
    parent_level1:= a;

    FOR level2 IN 1..number_of_tasks2 LOOP
           a:= a + 1;
           l_task_rec.pm_task_reference :=a;
           l_task_rec.task_name :='2 LEVEL '||a;
           l_task_rec.pm_parent_task_reference := parent_level1;
           l_tasks_in(a) := l_task_rec;
           parent_level2 := a;

           for level3 IN 1..number_of_tasks3 loop
                  a := a + 1;
                  l_task_rec.pm_task_reference := a;
                  l_task_rec.task_name :='3 LEVEL '||a;
                  l_task_rec.pm_parent_task_reference :=
parent_level2;
                  l_tasks_in(a) := l_task_rec;
                  parent_level3 := a;
                  for level4 IN 1..number_of_tasks4 loop
                         a := a + 1;
                         l_task_rec.pm_task_reference := a;
                         l_task_rec.task_name :='Fourth LEVEL
'||a;
                         l_task_rec.pm_parent_task_reference :=
parent_level3;
                         l_tasks_in(a) := l_task_rec;
                  end loop;

           end loop;

    END LOOP;
  end loop;
-----------------------
--INIT_CREATE_PROJECT
pa_project_pub.init_project;
-----------------------
--LOAD_PROJECT
pa_project_pub.load_project( p_api_version_number =>
l_api_version_number
                       ,p_return_status => l_return_status
```

```
                          ,p_created_from_project_id =>
l_created_from_project_id
                                ,p_project_name => l_project_name
                                ,p_description =>
l_project_description
                                ,p_pm_project_reference =>
l_pm_project_reference
                                ,p_pa_project_number =>
'rk-test-number'
                                ,p_carrying_out_organization_id =>
                                   l_carrying_out_organization_id
                                ,p_public_sector_flag  =>
l_public_sector_flag
                               ,p_customer_id => l_customer_id
                                ,p_project_status_code =>
l_project_status_code
                                ,p_start_date => l_start_date
                                ,p_completion_date =>
l_completion_date
                                ,p_actual_start_date =>
l_actual_start_date
                                ,p_actual_finish_date =>
l_actual_finish_date
                                ,p_early_start_date =>
l_early_start_date
                                ,p_early_finish_date =>
l_early_finish_date
                                ,p_late_start_date =>
l_late_start_date
                                ,p_late_finish_date =>
l_late_finish_date
                                ,p_distribution_rule =>
l_distribution_rule);
IF l_return_status != 'S'
THEN
    RAISE API_ERROR;
END IF;
-----------------------
--LOAD_KEY_MEMBER   (loop for multiple key members)

FOR i in 1..1 LOOP
pa_project_pub.load_key_member(  p_api_version_number =>
l_api_version_number
                           ,p_return_status => l_return_status
                           ,p_person_id =>
l_key_member_tbl(i).person_id
                           ,p_project_role_type  =>
                          l_key_member_tbl(i).project_role_type );
IF l_return_status != 'S'
```

```
THEN
    RAISE API_ERROR;
END IF;
END LOOP;
-----------------------
--LOAD_CLASS_CATEGORY  (loop for mulitple class categories-This
example has
-- only one )
FOR i IN 1..1 LOOP
pa_project_pub.load_class_category(
                          p_api_version_number =>
l_api_version_number
                     ,p_return_status => l_return_status
                     ,p_class_category => l_class_category
                     ,p_class_code => l_class_code );
IF l_return_status != 'S'
THEN
    RAISE API_ERROR;
END IF;
END LOOP;
-----------------------
--LOAD_TASK  (loop for multiple tasks)
FOR i IN 1..a LOOP
pa_project_pub.load_task( p_api_version_number =>
l_api_version_number
                    ,p_return_status  => l_return_status
                          ,p_pm_task_reference =>
l_tasks_in(i).pm_task_reference
                          ,p_task_name => l_tasks_in(i).task_name
                          ,p_pm_parent_task_reference =>
                             l_tasks_in(i).pm_parent_task_reference
                          ,p_task_start_date =>
l_tasks_in(i).task_start_date
                          ,p_task_completion_date =>
                             l_tasks_in(i).task_completion_date
                            ,p_actual_start_date =>
l_tasks_in(i).actual_start_date
                            ,p_actual_finish_date =>
l_tasks_in(i).actual_finish_date
                            ,p_early_start_date =>
l_tasks_in(i).early_start_date
                            ,p_early_finish_date =>
l_tasks_in(i).early_finish_date
                            ,p_late_start_date =>
l_tasks_in(i).late_start_date
                            ,p_late_finish_date =>
l_tasks_in(i).late_finish_date
                          ,p_address_id =>
l_tasks_in(i).address_id);
```

```
IF l_return_status != 'S'
THEN
    RAISE API_ERROR;
END IF;
END LOOP;
-----------------------
--EXECUTE_CREATE_PROJECT
pa_project_pub.execute_create_project(p_api_version_number =>
                                      l_api_version_number
                                     ,p_commit => l_commit
                                     ,p_init_msg_list => 'F'
                                     ,p_msg_count => l_msg_count
                                     ,p_msg_data => l_msg_data
                                     ,p_return_status => l_return_status
                                     ,p_pm_product_code =>
l_pm_product_code
                                     ,p_pa_project_id => l_project_id
                                     ,p_pa_project_number =>
l_pa_project_number);
IF l_return_status != 'S'
THEN
    RAISE API_ERROR;
END IF;
-----------------------
--FETCH_TASK
FOR l_task_index in 1..a LOOP
pa_project_pub.fetch_task( p_api_version_number =>
l_api_version_number
                          ,p_return_status => l_return_status
                          ,p_task_index => l_task_index
                          ,p_pa_task_id => l_task_id
                          ,p_pm_task_reference =>
l_pm_task_reference
                          ,p_task_return_status =>
l_task_return_status);
IF l_return_status != 'S'
OR l_task_return_status != 'S'
THEN
    RAISE API_ERROR;
END IF;
END LOOP;
-----------------------
--CLEAR_CREATE_PROJECT
pa_project_pub.clear_project;
IF l_return_status != 'S'
THEN
    RAISE API_ERROR;
END IF;
-----------------------
```

```
--HANDLE EXCEPTIONS
EXCEPTION
WHEN API_ERROR THEN
    for i in 1..l_msg_count loop
        pa_interface_utils_pub.get_messages (
                        p_msg_data            => l_msg_data
                          ,p_data => l_data
                       ,p_msg_count => l_msg_count
                         ,p_msg_index_out => l_msg_index_out );
        dbms_output.put_line ('error mesg '||l_data);

    end loop;
WHEN OTHERS THEN
    for i in 1..l_msg_count loop
        pa_interface_utils_pub.get_messages (
                        p_msg_count => l_msg_count
                       ,p_msg_data => l_msg_data
                          ,p_data => l_data
                         ,p_msg_index_out => l_msg_index_out);

         dbms_output.put_line ('error mesg '||l_data);
    end loop;
END ;
/
```

User–Defined Attribute APIs

You can use the user–defined attributes APIs to integrate user–defined attributes from an external system with Oracle Projects.

User–defined attributes enable you to capture unlimited information about projects and tasks to support the needs of your business. User–defined attributes are defined by the implementation team. They provide advanced project and task attribution with no coding, and feature a configurable user interface with complex validation. For more information about user–defined attributes, see User–Defined Attributes for Projects, *Oracle Projects Fundamentals* and Setting Up User–Defined Attributes, *Oracle Projects Implementation Guide*.

User–Defined Attribute Procedures

The procedures discussed in this section are listed below. The procedures are located in the public API package PA_PROJECT_PUB.

- LOAD_EXTENSIBLE_ATTRIBUTE
- LOAD_EXTENSIBLE_ATTRIBUTES

These procedures are called by the following load–execute–fetch procedures:

- Execute_Create_Project: page 3 – 43
- Execute_Update_Project: page 3 – 43

Global Constants

The package PA_PROJECT_PUB includes global constants, which are used for the parameter P_TRANSACTION_TYPE. The global constants are listed in the following table:

Constant	Description
G_CREATE_MODE	Creates the extensible attribute row
G_UPDATE_MODE	Updates an existing extensible attribute row

Table 3 – 8 Global constants (Page 1 of 2)

Constant	Description
G_DELETE_MODE	Deletes the extensible attribute row
G_SYNC_MODE	Creates / updates / deletes the extensible attribute row, as applicable

Table 3 – 8 Global constants (Page 2 of 2)

LOAD_EXTENSIBLE_ATTRIBUTE

This API loads a single attribute value for a given attribute group for the specified project and task. The following table shows the parameters for this procedure:

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	No	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
X_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_TRANSACTION_TYPE	IN	VARCHAR2	No	The mode of processing for a logical attribute group row. The value should correspond to the following constants in PA_PROJECT_PUB: G_DELETE_MODE, G_UPDATE_MODE, G_SYNC_MODE (which either creates or updates, as appropriate), and G_CREATE_MODE. Rows are processed in the order they are presented (that is, deletion first, followed by updates and synchronization, with creation last), in accordance with Apps standards.
P_TASK_ID	IN	NUMBER	1 (See Parameter Requirements : page 3 – 86)	Identifier of the task, if known. Required only if a task-level extensible attribute is provided.
P_TASK_REFERENCE	IN	VARCHAR2	1 (See Parameter Requirements : page 3 – 86)	Unique task reference, if task ID is unknown. Required only if a task-level extensible attribute is provided.

Name	Usage	Type	Req?	Description
P_ATTR_GRP_INTERNAL_NAME	IN	VARCHAR2	2 (See Parameter Requirements : page 3 – 86)	Internal name of the attribute group to which the current row belongs.
P_ATTR_GRP_ID	IN	NUMBER	2 (See Parameter Requirements : page 3 – 86)	Alternative to P_ATTR_GRP_INTERNAL_NAME
P_ATTR_GRP_ROW_INDEX	IN	NUMBER	Yes	Logical row identifier
P_ATTR_INTERNAL_NAME	IN	VARCHAR2	No	Internal name of the current row's attribute.
P_ATTR_VALUE_STR	IN	VARCHAR2	3 (See Parameter Requirements : page 3 – 86)	The value of the current row's attribute if its data type is String
P_ATTR_VALUE_NUM	IN	NUMBER	3 (See Parameter Requirements : page 3 – 86)	The value of the current row's attribute if its data type is Number
P_ATTR_VALUE_NUM_UOM	IN	VARCHAR2	No	The unit of measure selected to display number attributes
P_ATTR_VALUE_DATE	IN	DATE	3 (See Parameter Requirements : page 3 – 86)	The value for the current row's attribute if its data type is Date
P_ATTR_DISP_VALUE	IN	VARCHAR2	No	The value for the current row's aattribute (as a String, regardless of its data type) if the attribute has a value set with separate display and internal values (for example, value sets with validation type set to Independent or Table). In all other cases, use the preceding three columns.

Parameter Requirements

In the preceding parameter table, if the Required column contains a number, the following logic determines if a value is required:

Of the parameters that have the same number in the Required column, a value must be supplied for only one of the parameters. For example, P_TASK_ID and P_TASK_REFERENCE both have the number 1 in the Required column. A value must be supplied for either P_TASK_ID or P_TASK_REFERENCE.

LOAD_EXTENSIBLE_ATTRIBUTES

This is a bulk load API which loads the attribute values in a batch of 1000 attributes per API call. This procedure calls the LOAD_EXTENSIBLE_ATTRIBUTE API.

The following table shows the parameters for this procedure:

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	No	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
X_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_TRANSACTION_TYPE	IN	PA_VC_1000_10	No	The mode of processing for a logical attribute group row. The value should correspond to the following constants in PA_PROJECT_PUB: G_DELETE_MODE, G_UPDATE_MODE, G_SYNC_MODE (which either creates or updates, as appropriate), and G_CREATE_MODE. Rows are processed in the order they are presented (that is, deletion first, followed by updates and synchronization, with creation last), in accordance with Apps standards.
P_TASK_ID	IN	PA_NUM_1000_NUM	1 (See Parameter Requirements: page 3 – 86)	Identifier of the task, if known. Required only if a task-level extensible attribute is provided.
P_TASK_REFERENCE	IN	PA_VC_1000_150	1 (See Parameter Requirements: page 3 – 86)	Unique task reference, if task ID is unknown. Required only if a task-level extensible attribute is provided.
P_ATTR_GRP_INTERNAL_NAME	IN	PA_VC_1000_30	2 (See Parameter Requirements: page 3 – 86)	Internal name of the attribute group to which the current row belongs.
P_ATTR_GRP_ID	IN	PA_NUM_1000_NUM	2 (See Parameter Requirements: page 3 – 86)	Alternative to P_ATTR_GRP_INTERNAL_NAME
P_ATTR_GRP_ROW_INDEX	IN	PA_NUM_1000_NUM	Yes	Logical row identifier
P_ATTR_INTERNAL_NAME	IN	PA_VC_1000_30	No	Internal name of the current row's attribute.

Name	Usage	Type	Req?	Description
P_ATTR_VALUE_STR	IN	PA_VC_1000_150	3 (See Parameter Requirements: page 3 – 86)	The value of the current row's attribute if its data type is String
P_ATTR_VALUE_NUM	IN	PA_NUM_1000_NUM	3 (See Parameter Requirements: page 3 – 86)	The value of the current row's attribute if its data type is Number
P_ATTR_VALUE_NUM_UOM	IN	PA_VC_1000_30	No	The unit of measure selected to display number attributes
P_ATTR_VALUE_DATE	IN	PA_DATE_1000_DATE	3 (See Parameter Requirements: page 3 – 86)	The value for the current row's attribute if its data type is Date
P_ATTR_DISP_VALUE	IN	PA_VC_1000_150	No	The value for the current row's aattribute (as a String, regardless of its data type) if the attribute has a value set with separate display and internal values (for example, value sets with validation type set to Independent or Table). In all other cases, use the preceding three columns.

Using the User–Defined Attribute APIs

One feature of user–defined attributes is support for single– and multi–row attributes. This is an important concept to consider when you integrate user–defined attributes from an external system.

Single– and Multi–Row Attribute Groups

In this example, two attribute groups are defined for a project. The attribute groups are Project Complexity and Application Weightings.

Project Complexity is a single–row attribute group, which shows many records of information. Each record is displayed in a row in the table, with several columns across the page.

The following illustration shows how these groups appear in an entry screen.

Figure 3 – 3 Example of Attribute Groups in an Entry Screen

Integrating Single–Row Attribute Groups

The following table shows the data in the single–row attribute group *Project Complexity.*

Attribute	Value
AIM Project Type	AIM
Process Change Required	Low
System Size / Complexity	Medium

Table 3 – 9 (Page 1 of 2)

Attribute	Value
Customization Required	Medium
Complexity Score	[blank]

Table 3 – 9 (Page 2 of 2)

To load this information using a bulk approach with the API LOAD_EXTENSIBLE_ATTRIBUTES, a PL/SQL record is used to load each cell in the table. All the cells in a single row aredentified by using a common Attribute Row Identified.

> **Note:** Alternatively, the attribute/value pairs could be loaded one at a time using the LOAD_EXTENSIBLE_ATTRIBUTE API.

The following table illustrates how the Attribute Row Identifier puts the attributes into a single row.

PL/SQL Record Number	Attribute Row Identifier	Internal Attribute Group Name	Internal Attribute Name	Attribute Value (String)	Attribute Value (Number)	Attribute Value (Date)
1	1	Project Complexity	AIM Project Type	AIM	[blank]	[blank]
2	1	Project Complexity	Process Change Required	Low	[blank]	[blank]
3	1	Project Complexity	System Size/Complexity	Medium	[blank]	[blank]
4	1	Project Complexity	Customization Required	Medium	[blank]	[blank]
5	1	Project Complexity	Implementation Type	Phased	[blank]	[blank]
6	1	Project Complexity	Complexity Score	[blank]	[blank]	[blank]

Table 3 – 10 (Page 1 of 1)

Integrating Multi–Row Attribute Groups

The following table shows the data in the multi–row attribute group *Application Weightings.*

Product Family	Application Module	Default Weightings
Financials	Assets	0.4
Financials	General Ledger	0.6

Table 3 – 11 (Page 1 of 1)

To load the information shown here, a PL/SQL record is used to load each cell in the table above. All the cells in a single row can be identified by using a common Attribute Row Identified.

The following table shows the logical approach for loading this information using the LOAD_EXTENSIBLE_ATTRIBUTES bulk load API.

This example illustrates how the Attribute Row Identifier is used to group the attributes into a single row. Using the bulk load approach, you can load several attribute groups (both single– and multi–row) in one call to the API.

PL/SQL Record Number	Attribute Row Identifier	Internal Attribute Group Name	Internal Attribute Name	Attribute Value (String)	Attribute Value (Number)	Attribute Value (Date)
1	1	Application Weightings	Product Family	Financials	[blank]	[blank]
2	1	Application Weightings	Application Module	Assets	[blank]	[blank]
3	1	Application Weightings	Default Weighting	[blank]	0.4	[blank]
4	1	Application Weightings	Product Family	Financials	[blank]	[blank]
5	1	Application Weightings	Application Module	General Ledger	[blank]	[blank]
6	1	Application Weightings	Default Weighting	[blank]	0.6	[blank]

Table 3 – 12 (Page 1 of 1)

Example of Using the LOAD_EXTENSIBLE_ATTRIBUTE API

The following sample script shows how you can use the LOAD_EXTENSIBLE_ATTRIBUTE API to integrate a single attribute/value pair for a user–defined attribute group.

```
/*
Name: EATESTPACKAGE.SQL
Purpose: Package for the project amg api procedures' wrappers.
```

```
    */
create or replace package pa_EA_test as
    procedure create_project_EA(
     created_from_project_id number
    ,project_name varchar2
    );
end pa_EA_test;
/
CREATE OR REPLACE PACKAGE BODY PA_EA_TEST as
    procedure create_project_EA(
    created_from_project_id number
    ,project_name varchar2
    ) as
--This package is an example of how the LOAD_EXTENSIBLE_ATTRIBUTE API can be used
--to integrate a single attribute/value pair for a user-defined attribute group.

    --variables needed to create task hierarchy
    level1 number;
    level2 number;
    level3 number;
    a number := 0;
    m number := 0;
    parent_level1 varchar2(30);
    parent_level2 varchar2(30);
    parent_level3 varchar2(30);
    parent_level4 varchar2(30);
    parent_level5 varchar2(30);
    number_of_tasks1 number;
    number_of_tasks2 number;
    number_of_tasks3 number;
    number_of_tasks4 number;
    number_of_tasks5 number;
    number_of_tasks6 number;
    temp_msg_data varchar2(2000);

    --variables needed for api standard parameters
    l_api_version_number number := 1.0;
    l_commit varchar2(1) := 'T';
    l_return_status varchar2(1);
    l_init_msg_list varchar2(1) := 'T';
    l_msg_data varchar2(2000);
    l_msg_entity varchar2(100);
    l_msg_entity_index number;
    l_msg_index number;
```

```
l_encoded varchar2(1);
l_work_flow_started varchar2(1);
t1 varchar2(10);
t2 varchar2(100);
l_data varchar2(200);
t3 varchar2(2000);
l_msg_count number;
l_msg_index_out number;

--variables needed for oracle project specific parameters
l_created_from_project_id number;
l_pm_product_code varchar2(10);
l_number_of_task_levels number;
l_project_name varchar2(30);
l_project_number varchar2(80);
l_pm_project_reference varchar2(25);
l_project_status_code varchar2(30);
l_distribution_rule varchar2(30);
l_public_sector_flag varchar2(1);
l_carrying_out_organization_id number;
l_start_date date;
l_completion_date date;
l_actual_start_date date;
l_actual_finish_date date;
l_early_start_date date;
l_early_finish_date date;
l_late_start_date date;
l_late_finish_date date;
l_person_id number;
l_project_role_type varchar2(20);
l_class_category varchar2(30);
l_class_code varchar2(30);
l_project_id number(15);
l_pa_project_number varchar2(25);
l_project_description varchar2(250);
l_customer_id number;
l_project_relationship_code varchar2(30);
l_task_id number(15);
l_pm_task_reference varchar2(25);
l_task_index number;
project_loop number;
l_tasks_in pa_project_pub.task_in_tbl_type;
l_task_rec pa_project_pub.task_in_rec_type;
l_key_member_rec pa_project_pub.project_role_rec_type;
```

```
l_key_member_tbl pa_project_pub.project_role_tbl_type;
l_task_return_status varchar2(1);
l_short_name varchar2(10);
l_role_list_id number;
l_work_type_id number;
l_calendar_id number;
l_location_id number;
l_probability_member_id number;
l_project_value number;
l_opp_value_currency_code varchar2(15) := 'USD';
l_expected_approval_date date;
api_error exception;
l_org_member_rec pa_project_pub.project_role_rec_type;
l_org_member_tbl pa_project_pub.project_role_tbl_type;
l_task_version_id number;
l_encoded_msg varchar2(4000);
l_decoded_msg varchar2(4000);
l_final_msg varchar2(4000);
l_structure_type varchar2(25);
l_structure_version_name varchar2(25);
l_structure_version_id varchar2(25);
l_structure_description varchar2(150);
l_long_name varchar2(80);
v_time_before number;

l_ATTR_GRP_ROW_INDEX number;
l_ATTR_GRP_INTERNAL_NAME varchar2(15);
l_ATTR_INTERNAL_NAME varchar2(15);
l_ATTR_DISP_VALUE varchar2(15);


BEGIN

v_time_before := DBMS_UTILITY.get_time;

--PRODUCT RELATED DATA
l_pm_product_code := 'MSPROJECT';

--PROJECT DATA
l_created_from_project_id := created_from_project_id;
l_project_name := project_name;
l_project_number := project_name;
l_pm_project_reference := project_name;
l_project_description := project_name;
```

```
l_long_name := project_name;
l_project_status_code := '';
l_carrying_out_organization_id := 244;
l_start_date := '01-jan-00';
l_completion_date := '31-mar-05';
l_actual_start_date := '01-jan-01';
l_actual_finish_date := '01-apr-05';
l_early_start_date := '01-jan-01';
l_early_finish_date := '01-apr-05';
l_late_start_date := '01-jan-01';
l_late_finish_date := '01-APR-05';
l_role_list_id := 1000 ;
l_work_type_id := 10020;
l_calendar_id := 550;
l_location_id := 1;
l_probability_member_id := 1005;
l_project_value := 1000;
l_expected_approval_date := '31-mar-99';

--KEY MEMBERS DATA
l_key_member_rec.person_id := 53;
l_key_member_rec.project_role_type := 'PROJECT MANAGER';
l_key_member_tbl(1) := l_key_member_rec;

--CLASS CATEGORIES DATA
l_class_category := 'Product';
l_class_code := 'Non-classified';

-- EXTENSIBLE ATTRIBUTES DATA

l_ATTR_GRP_ROW_INDEX := 1;
l_ATTR_GRP_INTERNAL_NAME:= 'Project Complexity';
l_ATTR_INTERNAL_NAME := 'AIM Project Type';
l_ATTR_DISP_VALUE := 'AIM';

--TASKS DATA
--Set the number of tasks for every level (there are 6 levels)
number_of_tasks1 := 5;
number_of_tasks2 := 2;
number_of_tasks3 := 0;
number_of_tasks4 := 0;
number_of_tasks5 := 0;
number_of_tasks6 := 0;
```

```
a := 0;

for level1 in 1..number_of_tasks1 loop
 a:= a + 1;
 l_task_rec.pm_task_reference := a;
 l_task_rec.task_name := 'TOP LEVEL '|| a;
 l_task_rec.pm_parent_task_reference := '';
 l_task_rec.task_start_date := '01-jan-00';
 l_task_rec.task_completion_date := '31-mar-05';
 l_task_rec.actual_start_date := '01-JAN-01';
 l_task_rec.actual_finish_date := '01-APR-05';
 l_task_rec.early_start_date := '01-JAN-01';
 l_task_rec.early_finish_date := '01-APR-05';
 l_task_rec.late_start_date := '01-JAN-01';
 l_task_rec.late_finish_date := '01-APR-05';

 l_tasks_in(a) := l_task_rec;
 parent_level1 := a;

 FOR level2 IN 1..number_of_tasks2 LOOP
 a := a + 1;
 l_task_rec.pm_task_reference := a;
 l_task_rec.task_name := '2 LEVEL '|| a;
 l_task_rec.pm_parent_task_reference := parent_level1;
 l_tasks_in(a) := l_task_rec;
 parent_level2 := a;

 for level3 IN 1..number_of_tasks3 loop
 a := a + 1;
 l_task_rec.pm_task_reference := a;

 l_task_rec.task_name :='3 LEVEL '|| a;
 l_task_rec.pm_parent_task_reference := parent_level2;
 l_tasks_in(a) := l_task_rec;
 parent_level3 := a;

 for level4 IN 1..number_of_tasks4 loop
 a := a + 1;
 l_task_rec.pm_task_reference := a;
 l_task_rec.task_name := '4 LEVEL '|| a;
 l_task_rec.pm_parent_task_reference := parent_level3;
 l_tasks_in(a) := l_task_rec;

 for level5 IN 1..number_of_tasks5 loop
```

```
 a := a + 1;
 l_task_rec.pm_task_reference := a;
 l_task_rec.task_name := '5 LEVEL '|| a;
 l_task_rec.pm_parent_task_reference := parent_level4;
 l_tasks_in(a) := l_task_rec;

 for level6 IN 1..number_of_tasks6 loop
 a := a + 1;
 l_task_rec.pm_task_reference := a;
 l_task_rec.task_name := '6 LEVEL '|| a;
 l_task_rec.pm_parent_task_reference := parent_level5;
 l_tasks_in(a) := l_task_rec;
 end loop;--6th level
 end loop;--5th level
 end loop;--4th level
 end loop;--3rd level
 end loop;--2nd level
end loop;--1st level

-----------------------
dbms_output.put_line('Total tasks processed. '||l_tasks_in.count);

-----------------------
--INIT_CREATE_PROJECT

pa_project_pub.init_project;
-----------------------
--dbms_output.put_line('Before load_project');
--LOAD_PROJECT

pa_project_pub.load_project( p_api_version_number => l_api_version_number
 ,p_return_status => l_return_status
 ,p_created_from_project_id => l_created_from_project_id
 ,p_project_name => l_project_name
 ,p_long_name => l_long_name
 ,p_description => l_project_description
 ,p_pm_project_reference => l_pm_project_reference
 ,p_pa_project_number => l_project_number
 ,p_carrying_out_organization_id => l_carrying_out_organization_id
 ,p_public_sector_flag => l_public_sector_flag
 ,p_customer_id => l_customer_id
 ,p_project_status_code => l_project_status_code
 ,p_start_date => l_start_date
 ,p_completion_date => l_completion_date
```

```
 ,p_actual_start_date => l_actual_start_date
 ,p_actual_finish_date => l_actual_finish_date
 ,p_early_start_date => l_early_start_date
 ,p_early_finish_date => l_early_finish_date
 ,p_late_start_date => l_late_start_date
 ,p_late_finish_date => l_late_finish_date
 ,p_role_list_id => l_role_list_id
 ,p_work_type_id => l_work_type_id
 ,p_calendar_id => l_calendar_id
 ,p_location_id => l_location_id
 ,p_probability_member_id => l_probability_member_id
 ,p_project_value => l_project_value
 ,p_opp_value_currency_code => l_opp_value_currency_code
 ,p_expected_approval_date => l_expected_approval_date
 ,p_distribution_rule => l_distribution_rule);

if l_return_status != 'S' then
 raise api_error;
end if;
-----------------------
dbms_output.put_line('Before Loading Extensible Attributes');
pa_project_pub.load_extensible_attribute(
 p_api_version_number => l_api_version_number,
 x_return_status => l_return_status,
 P_ATTR_GRP_ROW_INDEX => l_ATTR_GRP_ROW_INDEX,
 P_ATTR_GRP_INTERNAL_NAME => l_ATTR_GRP_INTERNAL_NAME,
 P_ATTR_INTERNAL_NAME => l_ATTR_INTERNAL_NAME,
 P_ATTR_DISP_VALUE => l_ATTR_DISP_VALUE );

IF l_return_status != 'S'
THEN
 RAISE API_ERROR;
END IF;

-----------------------
--dbms_output.put_line('Before load_structure');
--LOAD_PROJECT

l_structure_type := 'FINANCIAL';
pa_project_pub.load_structure(
p_api_version_number => l_api_version_number
,p_return_status => l_return_status
,p_structure_type => l_structure_type
);
```

```
if l_return_status != 'S' then
 raise api_error;
end if;
-----------------------
--LOAD_KEY_MEMBER (loop for multiple key members)
pa_project_pub.load_key_member(
p_api_version_number => l_api_version_number
,p_return_status => l_return_status
,p_person_id => l_key_member_tbl(1).person_id
,p_project_role_type => l_key_member_tbl(1).project_role_type
);

IF l_return_status != 'S' THEN
 RAISE API_ERROR;
END IF;
-----------------------
--dbms_output.put_line('bef load task');
------------------------
--LOAD_TASK (loop for multiple tasks)
FOR i IN 1..a LOOP
 pa_project_pub.load_task(
 p_api_version_number => l_api_version_number
 ,p_return_status => l_return_status
 ,p_pm_task_reference => l_tasks_in(i).pm_task_reference
 ,p_task_name => l_tasks_in(i).task_name
 ,p_pm_parent_task_reference => l_tasks_in(i).pm_parent_task_reference
 ,p_task_start_date => l_tasks_in(i).task_start_date
 ,p_task_completion_date => l_tasks_in(i).task_completion_date
 ,p_actual_start_date => l_tasks_in(i).actual_start_date
 ,p_actual_finish_date => l_tasks_in(i).actual_finish_date
 ,p_early_start_date => l_tasks_in(i).early_start_date
 ,p_early_finish_date => l_tasks_in(i).early_finish_date
 ,p_late_start_date => l_tasks_in(i).late_start_date
 ,p_late_finish_date => l_tasks_in(i).late_finish_date
 ,p_address_id => l_tasks_in(i).address_id
 );

 IF l_return_status != 'S' THEN
 RAISE API_ERROR;
 END IF;
END LOOP;

--dbms_output.put_line('bef execute create project');
```

```
    -----------------------
    --EXECUTE_CREATE_PROJECT
    pa_project_pub.execute_create_project(
    p_api_version_number => l_api_version_number
    ,p_commit => l_commit
    ,p_init_msg_list => 'T'
    ,p_msg_count => l_msg_count
    ,p_msg_data => l_msg_data
    ,p_return_status => l_return_status
    ,p_workflow_started => l_work_flow_started
    ,p_pm_product_code => l_pm_product_code
    ,p_pa_project_id => l_project_id
    ,p_pa_project_number => l_pa_project_number
    );

    --dbms_output.put_line ('status '||l_return_status || ' msg count
'||l_msg_count);

    IF l_return_status in( 'E', 'U' ) THEN
     dbms_output.put_line( 'l_msg_data '||l_msg_data );
     dbms_output.put_line( 'Error count '||l_msg_count );
     l_msg_count := fnd_msg_pub.count_msg;

     FOR l_counter IN REVERSE 1..l_msg_count LOOP
     PA_UTILS.Get_Encoded_Msg(
     p_index => l_counter
     ,p_msg_out => l_encoded_msg);

     fnd_message.set_encoded(l_encoded_msg);
     l_decoded_msg := fnd_message.get;
     l_final_msg := l_final_msg || nvl(l_decoded_msg, l_encoded_msg);

     dbms_output.put_line( 'ERROR MESSAGE CODE: '|| l_counter|| ' :
'||l_encoded_msg );
     dbms_output.put_line( 'ERROR MESSAGE TEXT: '|| l_counter|| ' : '||
l_final_msg );
     END LOOP;
    ELSE
    dbms_output.put_line( 'l_return_status '|| l_return_status|| ' '||l_msg_data
);
    END IF;

    IF l_return_status != 'S' THEN
     RAISE API_ERROR;
```

```
END IF;

dbms_output.put_line ('Project Id '||l_project_id);
--dbms_output.put_line('bef execute fetch task');
-----------------------
--FETCH_TASK

FOR l_task_index in 1..a LOOP
 pa_project_pub.fetch_task(
 p_api_version_number => l_api_version_number
 ,p_return_status => l_return_status
 ,p_task_index => l_task_index
 ,p_pa_task_id => l_task_id
 ,p_pm_task_reference => l_pm_task_reference
 ,p_task_return_status => l_task_return_status
 );

 IF l_return_status != 'S' OR l_task_return_status != 'S' THEN
 dbms_output.put_line ('error text '|| SUBSTR (SQLERRM , 1 , 240));
 RAISE API_ERROR;
 END IF;
END LOOP;

--dbms_output.put_line('bef execute fetch str workplan');
-----------------------
--FETCH_TASK
pa_project_pub.fetch_structure_version(
p_api_version_number => l_api_version_number
,p_return_status => l_return_status
,p_structure_type => 'WORKPLAN'
,p_pa_structure_version_id => l_task_version_id
,p_struc_return_status => l_task_return_status
);

IF l_return_status != 'S' THEN
 dbms_output.put_line ('error text '|| SUBSTR (SQLERRM , 1 , 240));
 RAISE API_ERROR;
ELSE
 dbms_output.put_line (' Workplan Str ver id '||l_task_version_id );
END IF;

--dbms_output.put_line('bef execute fetch str financial');

pa_project_pub.fetch_structure_version(
```

```
    p_api_version_number => l_api_version_number
    ,p_return_status => l_return_status
    ,p_structure_type => 'FINANCIAL'
    ,p_pa_structure_version_id => l_task_version_id
    ,p_struc_return_status => l_task_return_status);

    IF l_return_status != 'S' THEN
     dbms_output.put_line ('error text '|| SUBSTR (SQLERRM , 1 , 240));
     RAISE API_ERROR;
    ELSE
     dbms_output.put_line (' Financial Str ver id '||l_task_version_id );
    END IF;
    ------------------------
    --CLEAR_CREATE_PROJECT
    pa_project_pub.clear_project;

    IF l_return_status != 'S' THEN
     RAISE API_ERROR;
    END IF;
    ------------------------
    --HANDLE EXCEPTIONS
    --COMMIT;

    DBMS_OUTPUT.put_line('Time elapsed in secs :' || (DBMS_UTILITY.get_time -
v_time_before)/(100));

    EXCEPTION
    WHEN API_ERROR THEN
    dbms_output.put_line( 'In Exception' );

    for i in 1..l_msg_count loop
     pa_interface_utils_pub.get_messages(
     p_data => l_data
     ,p_msg_index => i
     ,p_msg_count => l_msg_count
     ,p_msg_data => l_msg_data
     ,p_msg_index_out => l_msg_index_out
     );
     dbms_output.put_line ('error mesg '||l_data);
    end loop;
    end create_project_EA;
end pa_ea_test;
/
```

Example of Using the LOAD_EXTENSIBLE_ATTRIBUTES API

The following sample script shows how you can use the LOAD_EXTENSIBLE_ATTRIBUTES API to load a multi–row attribute group with three attributes, with both string and number attributes.

```
REM Using the LOAD_EXTENSIBLE_ATTRIBUTES bulk call
REM  Instructions to run this file to create a prjoject and add tasks to
FINANCIAL str.
REM  Change the following parameters
REM  l_created_from_project_id
REM    l_project_name
REM    l_project_number
REM    l_pm_project_reference
REM    l_project_description
REM    l_long_name
REM
REM   --Set the number of tasks for every level (there are 6 levels)
REM   number_of_tasks1 := 2;
REM   number_of_tasks2 := 3;
REM   number_of_tasks3 := 0;
REM   number_of_tasks4 := 0;
REM   number_of_tasks5 := 0;
REM   number_of_tasks6 := 0;
set serveroutput on;
execute dbms_application_info.set_client_info(458);
execute fnd_global.apps_initialize(1179, 20432, 275);
execute dbms_application_info.set_client_info(458);
--    PL/SQL example on how to create a project using the LOAD/EXECUTE/FETCH
--    mechanism
DECLARE
  --variables needed to create task hierarchy
  level1                                        NUMBER;
  level2                                        NUMBER;
  level3                                        NUMBER;
  a                                             NUMBER := 0;
  m                                             NUMBER := 0;
  parent_level1                                 VARCHAR2(30);
  parent_level2                                 VARCHAR2(30);
  parent_level3                                 VARCHAR2(30);
  parent_level4                                 VARCHAR2(30);
  parent_level5                                 VARCHAR2(30);
  number_of_tasks1                              NUMBER; --number of tasks/level
  number_of_tasks2                              NUMBER;
  number_of_tasks3                              NUMBER;
```

```
  number_of_tasks4                              NUMBER;
  number_of_tasks5                              NUMBER;
  number_of_tasks6                              NUMBER;

  temp_msg_data            VARCHAR2(2000);

  --variables needed for API standard parameters

  l_api_version_number                          NUMBER  :=1.0;
  l_commit                                      VARCHAR2(1):= 'F';
  l_return_status                               VARCHAR2(1);
  l_init_msg_list                               VARCHAR2(1);
  l_msg_data                                    VARCHAR2(2000);
  l_msg_entity                                  VARCHAR2(100);
  l_msg_entity_index                            NUMBER;
  l_msg_index                                   NUMBER;
  l_encoded                                     VARCHAR2(1);
  l_work_flow_started                           VARCHAR2(1);


t1  varchar2(10);
t2  varchar2(100);
l_data  varchar2(200);
t3     VARCHAR2(2000);
l_msg_count NUMBER;
l_msg_index_out NUMBER;


  --variables needed for Oracle Project specific parameters

  l_created_from_project_id                     NUMBER;
  l_pm_product_code                             VARCHAR2(10);
  l_number_of_task_levels                       NUMBER;
  l_project_name                                VARCHAR2(30);
  l_project_number                              VARCHAR2(80);
  l_pm_project_reference                        VARCHAR2(25);
  l_project_status_code                         VARCHAR2(30);
  l_distribution_rule                           VARCHAR2(30);
  l_public_sector_flag                          VARCHAR2(1);
  l_carrying_out_organization_id                NUMBER;
  l_start_date                                  DATE;
  l_completion_date                             DATE;
  l_actual_start_date                           DATE;
  l_actual_finish_date                          DATE;
```

```
  l_early_start_date                    DATE;
  l_early_finish_date                   DATE;
  l_late_start_date                     DATE;
  l_late_finish_date                    DATE;
  l_person_id                           NUMBER;
  l_project_role_type                   VARCHAR2(20);
  l_class_category                      VARCHAR2(30);
  l_class_code                          VARCHAR2(30);
  l_project_id                          NUMBER(15);
  l_pa_project_number                   VARCHAR2(25);
  l_project_description                 VARCHAR2(250);
  l_customer_id                         NUMBER;
  l_project_relationship_code           VARCHAR2(30);
  l_task_id                             NUMBER(15);
  l_pm_task_reference                   VARCHAR2(25);
  l_task_index                          NUMBER;
  project_loop                          NUMBER;
  l_tasks_in                           pa_project_pub.task_in_tbl_type;
  l_task_rec                           pa_project_pub.task_in_rec_type;
  l_ea_rec
pa_project_pub.PA_EXT_ATTR_ROW_TYPE;
  l_key_member_rec
pa_project_pub.project_role_rec_type;
  l_key_member_tbl
pa_project_pub.project_role_tbl_type;
  l_task_return_status                  VARCHAR2(1);
  l_short_name                          VARCHAR2(10);

   l_role_list_id                       NUMBER;
   l_work_type_id                       NUMBER;
   l_calendar_id                        NUMBER;
   l_location_id                        NUMBER;
   l_probability_member_id              NUMBER;
   l_project_value                      NUMBER;
   l_opp_value_currency_code          VARCHAR2(15) := 'USD';
   l_expected_approval_date             DATE;

  API_ERROR                             EXCEPTION;

  l_org_member_rec
pa_project_pub.project_role_rec_type;
  l_org_member_tbl
pa_project_pub.project_role_tbl_type;
  l_task_version_id                  NUMBER;
```

```
  l_encoded_msg                                   VARCHAR2(4000);
  l_decoded_msg                                   VARCHAR2(4000);
  l_final_msg                                     VARCHAR2(4000);

  l_structure_type                                VARCHAR2(25);
  l_structure_version_name                        VARCHAR2(25);
  l_structure_version_id                          VARCHAR2(25);
  l_structure_description                   VARCHAR2(150);
  l_long_name                                     VARCHAR2(80);
   v_time_before                            NUMBER;

  -- Extensible Attr variables;

  l_row_identifier_arr  pa_num_1000_num:= pa_num_1000_num();
  l_attr_group_int_name pa_vc_1000_30  := pa_vc_1000_30();
  l_attr_int_name       pa_vc_1000_30  := pa_vc_1000_30();
  l_attr_value_str      pa_vc_1000_150 := pa_vc_1000_150();
  l_attr_value_num      pa_num_1000_num:= pa_num_1000_num();
  l_attr_value_date     pa_date_1000_date:= pa_date_1000_date();


BEGIN
    v_time_before := DBMS_UTILITY.get_time;
PA_INTERFACE_UTILS_PUB.Set_Global_Info(
          p_api_version_number  => l_api_version_number
         ,p_responsibility_id   => 20432
         ,p_user_id             => 1179
         ,p_advanced_proj_sec_flag => 'Y'
         ,p_msg_count           => l_msg_count
         ,p_msg_data            => l_msg_data
         ,p_return_status       => l_return_status
        );

dbms_application_info.set_client_info(458);
for project_loop in 1..1 loop
--PRODUCT RELATED DATA
  l_pm_product_code                             :='MSPROJECT';

--PROJECT DATA
  l_created_from_project_id                 := 13086;
  l_project_name                                 := 'zk0425_11';
  l_project_number                                 := l_project_name;
  l_pm_project_reference                    := l_project_name;
  l_project_description                          := l_project_name;
```

```
  l_long_name                                    := 'Long name AMG project' ||
l_project_name;
  l_project_status_code                        := '';
  l_carrying_out_organization_id               :=244;
  l_start_date                                 :='01-jan-94';
  l_completion_date                            :='31-mar-15';
  l_actual_start_date                          :='01-jan-94';
  l_actual_finish_date                         :='01-apr-15';
  l_early_start_date                           :='01-jan-94';
  l_early_finish_date                          :='31-mar-15';
  l_late_start_date                            :='01-jan-94';
  l_late_finish_date                           :='31-mar-16';

   l_role_list_id                            :=1000 ;
   l_work_type_id                                :=10020;
   l_calendar_id                             :=550;
   l_location_id                                 :=1;
   l_probability_member_id                       :=1005;
   l_project_value                               :=1000;
   l_expected_approval_date                      := '31-mar-99';

--KEY MEMBERS DATA
m:= 1;

 l_person_id                              :='56';
 l_project_role_type                      :='PROJECT MANAGER';

--CLASS CATEGORIES DATA

  l_class_category                        :='Product';
  l_class_code                            :='Non-classified';

--TASKS DATA
  --Set the number of tasks for every level (there are 6 levels)
  number_of_tasks1 := 2;
  number_of_tasks2 := 1;
  number_of_tasks3 := 0;
  number_of_tasks4 := 0;
  number_of_tasks5 := 0;
  number_of_tasks6 := 0;

  a := 0;

  for level1 in 1..number_of_tasks1 loop
```

```
a:= a + 1;
l_task_rec.pm_task_reference                    :=a;
l_task_rec.task_name                            :='TOP LEVEL '||a;
l_task_rec.pm_parent_task_reference             :='';
l_task_rec.actual_start_date                    := '10-MAR-95';
l_task_rec.actual_finish_date                   := '06-JUL-10';
l_task_rec.early_start_date                     := '09-MAR-95';
l_task_rec.early_finish_date                    := '05-JUL-10';
l_task_rec.late_start_date                      := '09-MAR-95';
l_task_rec.late_finish_date                     := '05-JUL-10';
l_task_rec.scheduled_start_date                 := '01-jan-01';
l_task_rec.scheduled_finish_date                := '31-dec-05';

l_tasks_in(a):= l_task_rec;

parent_level1:= a;

        FOR level2 IN 1..number_of_tasks2 LOOP
        a:= a + 1;
        l_task_rec.pm_task_reference            :=a;
        l_task_rec.task_name                    :='2 LEVEL '||a;
        l_task_rec.scheduled_start_date         := '01-jan-02';
        l_task_rec.scheduled_finish_date        := '31-dec-07';
        l_task_rec.pm_parent_task_reference     := parent_level1;

                        l_tasks_in(a) := l_task_rec;
        parent_level2 := a;

        for level3 IN 1..number_of_tasks3 loop
                a := a + 1;
                l_task_rec.pm_task_reference         := a;
                l_task_rec.task_name                 :='3 LEVEL '||a;
                l_task_rec.pm_parent_task_reference  := parent_level2;
                l_tasks_in(a) := l_task_rec;
                parent_level3 := a;

                for level4 IN 1..number_of_tasks4 loop
                        a := a + 1;
                        l_task_rec.pm_task_reference           := a;
                        l_task_rec.task_name
                          :='Fourth LEVEL '||a;
                        l_task_rec.pm_parent_task_reference
                          := parent_level3;
```

```
                                l_tasks_in(a) := l_task_rec;

                          for level5 IN 1..number_of_tasks5 loop
                                    a := a + 1;
                                    l_task_rec.pm_task_reference
:= a;
                                    l_task_rec.task_name := 'Fifth LEVEL '||a;
                                    l_task_rec.pm_parent_task_reference :=
parent_level4;
                                    l_tasks_in(a) := l_task_rec;
                               for level6 IN 1..number_of_tasks6 loop
                                         a := a + 1;
                                         l_task_rec.pm_task_reference
:= a;
                                         l_task_rec.task_name := 'Sixth LEVEL
'||a;
                                         l_task_rec.pm_parent_task_reference :=
parent_level5;
                                         l_tasks_in(a) := l_task_rec;
                               end loop;    --6th level
                          end loop;         --5th level
                          end loop;                    --4th level
                  end loop;                           --3rd level
          END LOOP;                             --2nd level
    end loop;                                   --1st level
------------------------
dbms_output.put_line('Total tasks processed. '||l_tasks_in.count);

--can be used to exit this script and see how many tasks should have been
--created
------------------------
--INIT_CREATE_PROJECT

pa_project_pub.init_project;
------------------------
--dbms_output.put_line('Before load_project');

--LOAD_PROJECT

pa_project_pub.load_project( p_api_version_number       => l_api_version_number
                            ,p_return_status            => l_return_status
                            ,p_created_from_project_id =>
l_created_from_project_id
                            ,p_project_name             => l_project_name
```

```
                    ,p_long_name => l_long_name
                            ,p_description             => l_project_description
                           ,p_pm_project_reference    => l_pm_project_reference
                            ,p_pa_project_number   =>  l_project_number
                            ,p_carrying_out_organization_id =>
l_carrying_out_organization_id
                            ,p_public_sector_flag  => l_public_sector_flag
                            ,p_customer_id         => l_customer_id
                            ,p_project_status_code => l_project_status_code
                            ,p_start_date          => l_start_date
                            ,p_completion_date     => l_completion_date
                            ,p_actual_start_date   => l_actual_start_date
                            ,p_actual_finish_date  => l_actual_finish_date
                            ,p_early_start_date   => l_early_start_date
                            ,p_early_finish_date  => l_early_finish_date
                            ,p_late_start_date   => l_late_start_date
                            ,p_late_finish_date  => l_late_finish_date
                            ,p_role_list_id        => l_role_list_id
                            ,p_work_type_id      => l_work_type_id
                            ,p_calendar_id       => l_calendar_id
                            ,p_location_id       => l_location_id
                            ,p_probability_member_id=>l_probability_member_id
                            ,p_project_value       => l_project_value
                            ,p_opp_value_currency_code       =>
l_opp_value_currency_code
                           ,p_expected_approval_date=>l_expected_approval_date
                            ,p_distribution_rule => l_distribution_rule);

IF l_return_status != 'S'

THEN
        RAISE API_ERROR;
END IF;
-----------------------
--dbms_output.put_line('Before load_structure');

--LOAD_PROJECT

l_structure_type := 'FINANCIAL';

pa_project_pub.load_structure( p_api_version_number     => l_api_version_number
                              ,p_return_status    => l_return_status
                              ,p_structure_type       => l_structure_type
```

```
                              );

IF l_return_status != 'S'

THEN
        RAISE API_ERROR;
END IF;
------------------------
------------------------
--LOAD_CLASS_CATEGORY  (loop for mulitple class categories-This example has
-- only one )

FOR i IN 1..1 LOOP

pa_project_pub.load_class_category(
                          p_api_version_number        => l_api_version_number
                           ,p_return_status      => l_return_status
                           ,p_class_category     => l_class_category
                           ,p_class_code         => l_class_code );

IF l_return_status != 'S'
THEN
        RAISE API_ERROR;
END IF;

END LOOP;

dbms_output.put_line('bef load task');
-------------------------
--LOAD_TASK  (loop for multiple tasks)

FOR i IN 1..a LOOP

pa_project_pub.load_task( p_api_version_number          => l_api_version_number
                         ,p_return_status               => l_return_status
                         ,p_pm_task_reference           =>
                          l_tasks_in(i).pm_task_reference
                         ,p_task_name                   =>
                          l_tasks_in(i).task_name
                         ,p_pm_parent_task_reference    =>
                          l_tasks_in(i).pm_parent_task_reference
                         ,p_task_start_date             =>
                          l_tasks_in(i).task_start_date
                         ,p_task_completion_date        =>
```

```
                          l_tasks_in(i).task_completion_date
                         ,p_actual_start_date   =>
                          l_tasks_in(i).actual_start_date
                         ,p_actual_finish_date  =>
                          l_tasks_in(i).actual_finish_date
                         ,p_early_start_date    =>
                          l_tasks_in(i).early_start_date
                         ,p_early_finish_date   =>
                          l_tasks_in(i).early_finish_date
                         ,p_late_start_date   => l_tasks_in(i).late_start_date
                         ,p_late_finish_date  => l_tasks_in(i).late_finish_date
                  ,p_scheduled_start_date   => l_tasks_in(i).scheduled_start_date
                  ,p_scheduled_finish_date  => l_tasks_in(i).scheduled_finish_date
                          ,p_address_id                  =>
                           l_tasks_in(i).address_id);

IF l_return_status != 'S'

THEN
        RAISE API_ERROR;
END IF;

END LOOP;
------------------------
dbms_output.put_line('bef load ext attr');

--LOAD_EXTENSIBLE_ATTRIBUTE

  l_row_identifier_arr.extend;
  l_attr_group_int_name.extend;
  l_attr_int_name.extend;
  l_attr_value_str.extend;
  l_attr_value_num.extend;
  l_attr_value_date.extend;

  l_row_identifier_arr(1)       := 1;
  l_attr_group_int_name(1)      := 'Application Weightings';
  l_attr_int_name(1)            := 'Product Family';
  l_attr_value_str(1)           := 'Financials';

  l_row_identifier_arr.extend;
  l_attr_group_int_name.extend;
  l_attr_int_name.extend;
  l_attr_value_str.extend;
```

```
l_attr_value_num.extend;
l_attr_value_date.extend;

l_row_identifier_arr(2)       := 1;
l_attr_group_int_name(2)      := 'Application Weightings';
l_attr_int_name(2)            := 'Application Module';
l_attr_value_str(2)           := 'Assets';

l_row_identifier_arr.extend;
l_attr_group_int_name.extend;
l_attr_int_name.extend;
l_attr_value_str.extend;
l_attr_value_num.extend;
l_attr_value_date.extend;

l_row_identifier_arr(3)       := 1;
l_attr_group_int_name(3)      := 'Application Weightings';
l_attr_int_name(3)            := 'Default Weighting';
l_attr_value_num(3)           := 0.6;

l_row_identifier_arr.extend;
l_attr_group_int_name.extend;
l_attr_int_name.extend;
l_attr_value_str.extend;
l_attr_value_num.extend;
l_attr_value_date.extend;

l_row_identifier_arr(4)       := 2;
l_attr_group_int_name(4)      := 'Application Weightings';
l_attr_int_name(4)            := 'Product Family';
l_attr_value_str(4)           := 'Financials';

l_row_identifier_arr.extend;
l_attr_group_int_name.extend;
l_attr_int_name.extend;
l_attr_value_str.extend;
l_attr_value_num.extend;
l_attr_value_date.extend;

l_row_identifier_arr(5)       := 2;
l_attr_group_int_name(5)      := 'Application Weightings';
l_attr_int_name(5)            := 'Application Module';
l_attr_value_str(5)           := 'General Ledger';
```

```
  l_row_identifier_arr.extend;
  l_attr_group_int_name.extend;
  l_attr_int_name.extend;
  l_attr_value_str.extend;
  l_attr_value_num.extend;
  l_attr_value_date.extend;

  l_row_identifier_arr(6)       := 2;
  l_attr_group_int_name(6)      := 'Application Weightings';
  l_attr_int_name(6)            := 'Default Weighting';
  l_attr_value_num(6)           := 0.4;


dbms_output.put_line('bef load ext attr API CALL');

pa_project_pub.load_extensible_attributes(
                     p_api_version_number       => l_api_version_number
                     ,x_return_status      => l_return_status
                     ,P_ATTR_GRP_ROW_INDEX   => l_row_identifier_arr
                     ,P_ATTR_GRP_INTERNAL_NAME=> l_attr_group_int_name
                     ,P_ATTR_INTERNAL_NAME    => l_attr_int_name
                     ,P_ATTR_VALUE_STR   => l_attr_value_str
                     ,P_ATTR_VALUE_NUM   => l_attr_value_num
                     ,P_ATTR_VALUE_DATE  => l_attr_value_date
                     );
dbms_output.put_line('After load ext attr');

IF l_return_status != 'S'
THEN
        RAISE API_ERROR;
END IF;


--dbms_output.put_line(after load ext attr');

-------------------------

--dbms_output.put_line('bef execute create project');

-----------------------

--EXECUTE_CREATE_PROJECT

pa_project_pub.execute_create_project(p_api_version_number =>
```

```
                                    l_api_version_number
                                   ,p_commit          => l_commit
                                   ,p_init_msg_list   => 'F'
                                   ,p_msg_count        => l_msg_count
                                   ,p_msg_data                => l_msg_data
                                   ,p_return_status   => l_return_status
                                   ,p_workflow_started => l_work_flow_started
                                   ,p_pm_product_code => l_pm_product_code
                                   ,p_pa_project_id   => l_project_id
                                   ,p_pa_project_number =>
                                    l_pa_project_number
       );

--dbms_output.put_line ('status '||l_return_status || ' msg count
'||l_msg_count);

   IF l_return_status in( 'E', 'U' )

   THEN
      dbms_output.put_line( 'l_msg_data '||l_msg_data );
      dbms_output.put_line( 'Error count '||l_msg_count );

      l_msg_count := fnd_msg_pub.count_msg;

      FOR l_counter IN REVERSE 1..l_msg_count LOOP

        PA_UTILS.Get_Encoded_Msg(p_index    => l_counter,
                    p_msg_out  => l_encoded_msg);

        fnd_message.set_encoded(l_encoded_msg);
        l_decoded_msg := fnd_message.get;

        l_final_msg := l_final_msg || nvl(l_decoded_msg, l_encoded_msg);

        dbms_output.put_line( 'ERROR MESSAGE CODE: '|| l_counter|| ' :
'||l_encoded_msg  );
        dbms_output.put_line( 'ERROR MESSAGE TEXT: '|| l_counter|| ' : '||
l_final_msg  );
      END LOOP;

   ELSE
      dbms_output.put_line( 'l_return_status '|| l_return_status|| '
'||l_msg_data );
   END IF;
```

```
IF l_return_status != 'S'
THEN
        RAISE API_ERROR;
END IF;
dbms_output.put_line ('Project Id '||l_project_id);

--dbms_output.put_line('bef execute fetch task');
-----------------------
--FETCH_TASK
FOR l_task_index in 1..a LOOP

pa_project_pub.fetch_task(       p_api_version_number   => l_api_version_number
                                ,p_return_status        => l_return_status
                                ,p_task_index           => l_task_index
                    ,p_pa_task_id           => l_task_id
                                ,p_pm_task_reference    => l_pm_task_reference
                               ,p_task_return_status   => l_task_return_status);

IF l_return_status != 'S'
OR l_task_return_status != 'S'
THEN
       dbms_output.put_line ('error text '|| SUBSTR (SQLERRM , 1 , 240));
        RAISE API_ERROR;

END IF;

END LOOP;



--dbms_output.put_line('bef execute fetch str workplan');

-----------------------

--FETCH_TASK

pa_project_pub.fetch_structure_version(
                                 p_api_version_number   => l_api_version_number
                                ,p_return_status        => l_return_status
                                ,p_structure_type       => 'WORKPLAN'
                                ,p_pa_structure_version_id => l_task_version_id
                                ,p_struc_return_status   =>
l_task_return_status);
```

```
IF l_return_status != 'S'
THEN
       dbms_output.put_line ('error text '|| SUBSTR (SQLERRM , 1 , 240));
        RAISE API_ERROR;
ELSE
    dbms_output.put_line (' Workplan Str ver id '||l_task_version_id );
END IF;

--dbms_output.put_line('bef execute fetch str financial');
pa_project_pub.fetch_structure_version(
                                p_api_version_number   => l_api_version_number
                                ,p_return_status        => l_return_status
                                ,p_structure_type       => 'FINANCIAL'
                                ,p_pa_structure_version_id => l_task_version_id
                                ,p_struc_return_status   =>
l_task_return_status);

IF l_return_status != 'S'
THEN
       dbms_output.put_line ('error text '|| SUBSTR (SQLERRM , 1 , 240));
        RAISE API_ERROR;
ELSE
    dbms_output.put_line (' Financial Str ver id '||l_task_version_id );
END IF;
------------------------
--CLEAR_CREATE_PROJECT

pa_project_pub.clear_project;

IF l_return_status != 'S'
THEN
        RAISE API_ERROR;
END IF;
------------------------
--HANDLE EXCEPTIONS
end loop;

    DBMS_OUTPUT.put_line (
       'Time elapsed in secs :' ||
          (DBMS_UTILITY.get_time - v_time_before) / (100)
    );

EXCEPTION
```

```
WHEN API_ERROR THEN

dbms_output.put_line( 'In Exception'||sqlerrm );
    for i in 1..l_msg_count loop

        pa_interface_utils_pub.get_messages (
                        p_encoded        => FND_API.G_TRUE,
                        p_data           => l_data
                       ,p_msg_index      => i
                       ,p_msg_count      => l_msg_count
                       ,p_msg_data       => l_msg_data
                       ,p_msg_index_out  => l_msg_index_out );
      dbms_output.put_line ('error mesg :'||l_data);

    end loop;
END ;
/
```

Structure APIs

The structure APIs enable you to use an external system to create and change structure versions. The structure APIs include:

- CHANGE_STRUCTURE_STATUS
- BASELINE_STRUCTURE
- LOAD_STRUCTURE
- DELETE_STRUCTURE_VERSION
- FETCH_STRUCTURE_VERSION

CHANGE_STRUCTURE_STATUS

Use this PL/SQL procedure to publish, submit, rework, reject, or approve a structure and thereby change its status code. Valid status codes are:

- STRUCTURE_WORKING
- STRUCTURE_PUBLISHED
- STRUCTURE_SUBMITTED
- STRUCTURE_REJECTED
- STRUCTURE_APPROVED

The following table shows the parameters for this procedure.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default =F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default =F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_STRUCTURE_VERSION_ID	IN	NUMBER	Yes	The unique identifier of the structure
P_PA_PROJECT_ID	IN	NUMBER	Yes	The unique identifier of the project

Name	Usage	Type	Req?	Description
P_STATUS_CODE	IN	VARCHAR2	Yes	One of the valid structure status codes: STRUCTURE_WORKING, STRUCTURE_PUBLISHED, STRUCTURE_SUBMITTED, STRUCTURE_REJECTED, STRUCTURE_APPROVED
P_PUBLISHED_STRUCT_VER_ID	OUT	NUMBER		The unique identifier of the published structure version

BASELINE_STRUCTURE

BASELINE_STRUCTURE is a PL/SQL procedure to baseline a structure version

The following table shows the parameters for this procedure.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default =F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default =F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_STRUCTURE_VERSION_ID	IN	NUMBER	Yes	The unique identifier of the structure
P_PA_PROJECT_ID	IN	NUMBER	Yes	The unique identifier of the project

LOAD_STRUCTURE

LOAD_STRUCTURE is a Load–Execute–Fetch procedure used to load structure data.

The following table shows the parameters for this procedure.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default =F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PA_PROJECT_ID	IN	NUMBER	No	The unique identifier of the project

Name	Usage	Type	Req?	Description
P_STRUCTURE_VERSION_NAME	IN	VARCHAR2	No	The name of the structure version
P_STRUCTURE_VERSION_ID	IN	NUMBER	No	The unique identifier of the structure
P_DESCRIPTION	IN	VARCHAR2	No	The unique identifier of the structure

DELETE_STRUCTURE_VERSION

DELETE_STRUCTURE_VERSION is a PL/SQL procedure used to delete a structure version from Oracle Projects.

The following table shows the parameters for this procedure.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default =F)
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default =F)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_STRUCTURE_VERSION_ID	IN	NUMBER	Yes	The unique identifier of the structure
P_RECORD_VERSION_NUMBER	OUT	NUMBER		The unique identifier of the published structure version

FETCH_STRUCTURE_VERSION

FETCH_STRUCTURE_VERSION is a Load–Execute–Fetch procedure that returns structure version IDs of workplan and financial structures.

The following table shows the parameters for this procedure.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default =F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_STRUCTURE_TYPE	IN	VARCHAR2	No	Structure type

Name	Usage	Type	Req?	Description
P_STRUCTURE_VERSION_ID	OUT	NUMBER	Yes	The unique identifier of the structure
P_STRUC_RETURN_STATUS	OUT	VARCHAR2		Structure status

Resource APIs

You can keep track of and organize both labor and non–labor resources using the system that you prefer. Then, use the resource APIs to export your resource lists and the resources they include to Oracle Projects. Oracle Projects updates its resource information accordingly. As your resources and resource lists change, update the information in your system and periodically synchronize the two systems.

> **Note:** When you call any resource API that requires a resource list identifier, pass either the P_RESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID parameter to identify the resource list. When you call any resource API that requires a resource identifier, pass either the P_RESOURCE_ALIAS or the P_RESOURCE_LIST_MEMBER_ID parameter to identify the resource.

Resource API Views

The following table lists the views that provide parameter data for the resource APIs. For detailed description of the views, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

View	Description
PA_AMG_RESOURCE_INFO_V	You can customize this view to retrieve additional information about resource list members, such as cost rates or overtime rates.
PA_EMPLOYEES_RES_V	Displays information about all employees defined in your human resources application. You can define any employee returned by this view as a resource in Oracle Projects.
PA_EVENT_TYPES_RES_V	Displays event types defined in Oracle Projects. You can define any event type returned by this view as a resource in Oracle Projects.
PA_EXPEND_CATEGORIES_RES_V	Displays expenditure categories defined in Oracle Projects. You can define any expenditure category returned by this view as a resource in Oracle Projects.

Table 3 – 13 Resource API views (Page 1 of 2)

View	Description
PA_EXPENDITURE_TYPES_RES_V	Displays expenditure types defined in Oracle Projects. You can define any expenditure type returned by this view as a resource in Oracle Projects.
PA_JOBS_RES_V	Displays information about all the jobs defined in your human resources application. You can define any job returned by this view as a resource in Oracle Projects.
PA_LOWEST_LEVEL_RESOURCES_V	Retrieves Oracle Projects identification codes and names for resource lists and lowest–level resource list members.
PA_ORGANIZATIONS_RES_V	Displays information about the organizations defined in your human resources application. You can define any organization returned by this view as a resource in Oracle Projects.
PA_PROJ_ORG_STRUCTURES_V	Retrieves the organization hierarchy
PA_QRY_RESOURCE_LISTS_V	Retrieves resource lists defined in Oracle Projects
PA_QUERY_RES_LIST_MEMBERS_V	Retrieves members of a resource list defined in Oracle Projects
PA_RESOURCE_LIST_GROUPS_V	Retrieves resource groups in a resource list defined in Oracle Projects
PA_RESOURCE_LIST_V	Retrieves resource lists defined in Oracle Projects
PA_RESOURCE_TYPES_ACTIVE_V	Retrieves active resource types defined in Oracle Projects
PA_REVENUE_CATEGORIES_RES_V	Displays revenue categories defined in Oracle Projects. You can define any revenue category returned by this view as a resource in Oracle Projects.
PA_VENDORS_RES_V	Displays information about vendors defined in Oracle Purchasing. You can define any vendor returned by this view as a resource in Oracle Projects.

Table 3 – 13 Resource API views (Page 2 of 2)

Resource API Procedures

- Resource List and Resource List Member Procedures
  - ADD_RESOURCE_LIST_MEMBER: page 3 – 127
  - CREATE_RESOURCE_LIST: page 3 – 128
  - DELETE_RESOURCE_LIST: page 3 – 130
  - DELETE_RESOURCE_LIST_MEMBER: page 3 – 131
  - SORT_RESOURCE_LIST_MEMBERS: page 3 – 132
  - UPDATE_RESOURCE_LIST: page 3 – 132
  - UPDATE_RESOURCE_LIST_MEMBER: page 3 – 134
- Load–Execute–Fetch Procedures
  - CLEAR_CREATE_RESOURCE_LIST: page 3 – 135
  - CLEAR_UPDATE_MEMBERS: page 3 – 135
  - EXEC_CREATE_RESOURCE_LIST: page 3 – 136
  - EXEC_UPDATE_RESOURCE_LIST: page 3 – 136
  - FETCH_MEMBERS: page 3 – 136
  - FETCH_RESOURCE_LIST: page 3 – 137
  - INIT_CREATE_RESOURCE_LIST: page 3 – 137
  - INIT_UPDATE_MEMBERS: page 3 – 137
  - LOAD_MEMBERS: page 3 – 138
  - LOAD_RESOURCE_LIST: page 3 – 139

Resource API Procedure Definitions

This section contains description of the resource APIs, including business rules and parameters.

ADD_RESOURCE_LIST_MEMBER

ADD_RESOURCE_LIST_MEMBER is a PL/SQL procedure that adds a resource member to an existing resource list.

Business Rules

1. Calling modules can pass either the RESOURCE_LIST_NAME or the RESOURCE_LIST_ID.
2. If the calling modules pass both RESOURCE_LIST_NAME and RESOURCE_LIST_ID, the API uses the latter.
3. If the resource list is grouped, you must pass a valid resource group alias.
4. The value for P_RESOURCE_ATTR_VALUE must correspond to the value for P_RESOURCE_TYPE. For example, the person identification code for P_RESOURCE_ATTR_VALUE must be valid if P_RESOURCE_TYPE equals EMPLOYEE.
5. If the calling module passes information for both RESOURCE_GROUP and RESOURCE_MEMBER parameters to this API, the API first verifies that the resource group exists. If the resource group does not exist, the API creates the resource group and then creates the resource.
6. If a given resource member already exists, this API does not return an error. Instead, it returns a successful return status and the RESOURCE_LIST_MEMBER_ID of the existing resource member.

   Note: Because you can store only one transaction attribute for a given resource, this API accepts only a single RESOURCE_ATTR_VALUE, which may hold PERSON_ID, JOB_ID, and so on.

The following table shows the parameters for ADD_RESOURCE_LIST_MEMBER.

Name	Identifier of the task, if known. Required only if task-level extensible attribute is prpovided.	Usage	Type	Req?	Description
P_COMMIT		IN	VARCHAR2(1)	No	API standard
P_INIT_MSG_LIST		IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER		IN	NUMBER	Yes	API standard
P_RESOURCE_LIST_NAME		IN	VARCHAR2(30)	No	Name of resource list
P_RESOURCE_LIST_ID		IN	NUMBER	No	The identification code of the resource list
P_RESOURCE_GROUP_ALIAS		IN	VARCHAR2(30)	No	Alias of the resource group
P_RESOURCE_GROUP_NAME		IN	VARCHAR2(80)	No	Name of the resource group
P_RESOURCE_TYPE_CODE		IN	VARCHAR2(30)	Yes	Type code of the resource
P_RESOURCE_ATTR_VALUE		IN	VARCHAR2(80)	Yes	Attribute value of the resource
P_RESOURCE_ALIAS		IN	VARCHAR2(30)	Yes	Alias of the resource member
P_SORT_ORDER		IN	NUMBER	No	Sort order of the resource member
P_ENABLED_FLAG		IN	VARCHAR2(1)	No	Enabled flag of the resource member (default = 'Y')
P_RESOURCE_LIST_MEMBER_ID		OUT	NUMBER		The identification code of the resource member on a specific resource list
P_MSG_COUNT		OUT	NUMBER		API standard
P_MSG_DATA		OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS		OUT	VARCHAR2(1)		API standard

CREATE_RESOURCE_LIST

CREATE_RESOURCE_LIST is a PL/SQL procedure that creates a resource list and optionally creates the resource list members.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes: page 2 – 19.

Business Rules

- Valid values for P_GROUP_RESOURCE_TYPE are EXPENDITURE_CATEGORY, REVENUE_CATEGORY, ORGANIZATION, and NONE.
- The resource list name must be unique.

- If calling programs pass the P_MEMBER_TBL (optional), this API creates the relevant resource list member records.
- If your resource list is grouped, you must pass a valid resource group alias.
- The value for P_RESOURCE_ATTR_VALUE must correspond with the value for P_RESOURCE_TYPE. For example, the person identification code for P_RESOURCE_ATTR_VALUE must be valid if P_RESOURCE_TYPE equals EMPLOYEE.
- If the value for GROUP_RESOURCE_TYPE is NONE, this API will ignore resource group IN parameters.
- If you do not specify the resource group alias, the group resource type must be NONE.

The following table shows the parameters for CREATE_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RESOURCE_LIST_REC	IN	RECORD		
RESOURCE_LIST_NAME	IN	VARCHAR2(80)	Yes	Name of resource list
DESCRIPTION	IN	VARCHAR2(255)	No	Description of resource list
GROUP_RESOURCE_TYPE	IN	VARCHAR2(30)	Yes	Type of resource group
START_DATE	IN	DATE	No	Start date of resource list
END_DATE	IN	DATE	No	End date of resource list
P_RESOURCE_LIST_OUT_REC	OUT	RECORD		API standard
RESOURCE_LIST_ID	OUT	NUMBER		The reference code that uniquely identifies the resource list
RETURN_STATUS	OUT	VARCHAR2(1)		Return status of specific resource list
P_MEMBER_TBL	IN	TABLE OF RECORD		
RESOURCE_GROUP_ALIAS	IN	VARCHAR2(30)	No	Alias of resource group
RESOURCE_GROUP_NAME	IN	VARCHAR2(80)	No	Name of resource group
RESOURCE_TYPE_CODE	IN	VARCHAR2(30)	Yes	Type code of resource member
RESOURCE_ATTR_VALUE	IN	VARCHAR2(80)	Yes	Attribute value of resource member

Name	Usage	Type	Req?	Description
RESOURCE_ALIAS	IN	VARCHAR2(30)	Yes	Alias of resource member
SORT_ORDER	IN	NUMBER	No	Sort order of resource member
ENABLED_FLAG	IN	VARCHAR2(1)	No	Enabled flag for resource member (default = 'Y')
P_MEMBER_OUT_TBL	OUT	TABLE OF RECORD		
RESOURCE_LIST_MEMBER_ID	OUT	NUMBER		The identification code of resource member on a specific resource list
RETURN_STATUS	OUT	VARCHAR2(1)		Return status of specific resource member

DELETE_RESOURCE_LIST

DELETE_RESOURCE_LIST is a PL/SQL procedure that deletes a given resource list.

Business Rules

- Calling modules can pass either the P_RESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID.
- If calling modules pass both P_RESOURCE_LIST_NAME and the P_RESOURCE_LIST_ID, this API uses the latter.
- You cannot delete a resource list if:
  - You summarize project actuals by that resource list.
  - A budget uses that resource list.
  - The list contains resource list members.

The following table shows the parameters for DELETE_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_RESOURCE_LIST_NAME	IN	VARCHAR2(30)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	Identification code of the resource list
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard

Name	Usage	Type	Req?	Description
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
X_ERR_CODE	IN OUT	NUMBER		The error handling code
X_ERR_STAGE	IN OUT	VARCHAR2		Error message text

DELETE_RESOURCE_LIST_MEMBER

DELETE_RESOURCE_LIST_MEMBER is a PL/SQL procedure that deletes a given resource list member.

Business Rules

- Calling modules can pass either the P_RESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID. Calling modules can also pass the P_ALIAS or the P_ALIAS_MEMBER_ID.
- If the calling modules pass both P_RESOURCE_LIST_NAME and the P_RESOURCE_LIST_ID, this API uses the latter.
- You cannot delete a resource list member if:
  - You summarize project actuals by that resource list member.
  - A budget uses that resource list member.

The following table shows the parameters for DELETE_RESOURCE_LIST_MEMBER.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RESOURCE_LIST_NAME	IN	VARCHAR2(30)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	Identification code of the resource list
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	Alias of the resource list
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	Identification code of the resource list member
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
X_ERR_CODE	IN OUT	NUMBER		The error handling code
X_ERR_STAGE	IN OUT	VARCHAR2		Error message text

SORT_RESOURCE_LIST_MEMBERS

SORT_RESOURCE_LIST_MEMBERS is a PL/SQL procedure that updates the sort order for resource members in a given resource list.

Business Rules

- Calling modules can pass either the P_RESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID.
- If the calling modules pass both the P_RESOURCE_LIST_NAME and the P_RESOURCE_LIST_ID, this API uses the latter.
- If you specify a resource group alias, this API sorts only resources below that resource group. Otherwise, this API sorts all resources in the resource list.
- You can sort resources by alias or resource name. Valid values for P_SORT_BY PARAMETER are ALIAS and RESOURCE_NAME.

The following table shows the parameters for SORT_RESOURCE_LIST_MEMBERS.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_RESOURCE_LIST_NAME	IN	VARCHAR2(30)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	Identification code of the resource list
P_RESOURCE_GROUP_ALIAS	IN	VARCHAR2(30)	No	Alias of the resource group
P_SORT_BY	IN	VARCHAR2(30)	Yes	Sort-by code
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

UPDATE_RESOURCE_LIST

UPDATE_RESOURCE_LIST is a PL/SQL procedure that updates an existing resource list, including updating existing or adding new resource list members.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes: page 2 – 19.

Business Rules

- Calling modules can pass either the RESOURCE_LIST_NAME or the RESOURCE_LIST_ID.
- If the calling modules pass both the RESOURCE_LIST_NAME and the RESOURCE_LIST_ID, this API uses the latter.
- You cannot change GROUPED_BY_TYPE if the resource list already contains active members.
- You can change the following fields at any time:
  - RESOURCE LIST NAME
  - DESCRIPTION
  - START DATE
  - END DATE
- You must enter a unique new resource list name.
- You can update existing or add new resource list members by including the member records in the MEMBER_TBL. If a resource list member already exists, you can update the following fields:
  - ALIAS. Specify the P_NEW_ALIAS.
  - SORT_ORDER. Specify the P_SORT_ORDER.

  Note: The alias must be unique within a resource group.

The following table shows the parameters for UPDATE_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RESOURCE_LIST_NAME	IN	VARCHAR2(30)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	Identification code of the resource list

Name	Usage	Type	Req?	Description
P_NEW_LIST_NAME	IN	VARCHAR2(30)	No	New name of the existing resource list
P_GROUPED_BY_TYPE	IN	VARCHAR2(30)	No	GROUP_BY_TYPE of the resource list
P_DESCRIPTION	IN	VARCHAR2(80)	No	Description
P_START_DATE	IN	DATE	No	Start date of the resource list
P_END_DATE	IN	DATE	No	End date of the resource list
P_MEMBER_TBL	IN	TABLE OF RECORD		
RESOURCE_GROUP_ALIAS	IN	VARCHAR2(30)	No	Alias of the resource group
RESOURCE_GROUP_NAME	IN	VARCHAR2(80)	No	Name of the resource group
RESOURCE_TYPE_CODE	IN	VARCHAR2(30)	Yes	Type code of the resource member
RESOURCE_ATTR_VALUE	IN	VARCHAR2(80)	Yes	Attribute value of the resource member
RESOURCE_ALIAS	IN	VARCHAR2(30)	Yes	Alias of the resource member
SORT_ORDER	IN	NUMBER	No	Sort order of the resource member
RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	Identification code of the resource list member
NEW_ALIAS MEMBER	IN	VARCHAR2(30)	No	New alias of the resource
ENABLED_FLAG	IN	VARCHAR2(1)	No	Enabled flag of the resource member (default = 'Y')
P_MEMBER_OUT_TBL	OUT	TABLE OF RECORD		
RESOURCE_LIST_MEMBER_ID	OUT	NUMBER		Identification code of the resource list member
RETURN_STATUS	OUT	VARCHAR2(1)		Return status of a specific resource list member

UPDATE_RESOURCE_LIST_MEMBER

UPDATE_RESOURCE_LIST_MEMBER is a PL/SQL procedure that updates the alias and enables or disables the resource list members.

Business Rules

- Calling modules can pass either the P_RESOURCE_LIST_NAME or P_RESOURCE_LIST_ID.
- If the calling modules pass both the P_RESOURCE_LIST_NAME and the P_RESOURCE_LIST_ID, this API uses the latter.

- You can use the P_ENABLED_FLAG to enable or disable a resource member. If the parameter value is passed as NULL or something other than Y, the column value remains the same.

Note: The alias must be unique within a resource group.

The following table shows the parameters for UPDATE_RESOURCE_LIST_MEMBER.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_RESOURCE_LIST_NAME	IN	VARCHAR2(30)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	Identification code of the resource list
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	Alias of the resource member
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	Identification code of the resource list member
P_NEW_ALIAS	IN	VARCHAR2(30)	No	New alias of the resource member
P_SORT_ORDER	IN	NUMBER	No	Sort order of the resource member
P_ENABLED_FLAG	IN	VARCHAR2(1)	No	Enabled flag of the resource member (default = 'Y')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

CLEAR_CREATE_RESOURCE_LIST

CLEAR_CREATE_RESOURCE_LIST is a Load–Execute–Fetch procedure used to clear the global data structures set up during the Initialize step.

CLEAR_UPDATE_MEMBERS

CLEAR_UPDATE_MEMBERS is a Load–Execute–Fetch procedure used to clear the global data structures that were set up during the Initialize step for the Load–Execute–Fetch update APIs.

EXEC_CREATE_RESOURCE_LIST

EXEC_CREATE_RESOURCE_LIST is a Load–Execute–Fetch procedure used to execute the composite API CREATE_RESOURCE_LIST.

The following table shows the parameters for EXEC_CREATE_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard

EXEC_UPDATE_RESOURCE_LIST

EXEC_UPDATE_RESOURCE_LIST is a Load–Execute–Fetch procedure used to execute the composite API UPDATE_RESOURCE_LIST.

The following table shows the parameters for EXEC_UPDATE_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RETURN_STATUS	OUt	VARCHAR2(1)		API standard
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard

FETCH_MEMBERS

FETCH_MEMBERS is a Load–Execute–Fetch procedure used to fetch resource members from the global output structure for resource list members.

The following table shows the parameters for FETCH_MEMBERS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_MEMBER_INDEX	IN	NUMBER		Member Index (default = 1)
P_RESOURCE_LIST_MEMBER_ID	OUT	NUMBER		Identification code of the resource list member
P_MEMBER_RETURN_STATUS	OUT	VARCHAR2(1)		Return status of the specific resource list member

FETCH_RESOURCE_LIST

FETCH_RESOURCE_LIST is a Load–Execute–Fetch procedure used to fetch one resource list identifier at a time from the global output structure for resource lists.

The following table shows the parameters for FETCH_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_RESOURCE_LIST_ID	OUT	NUMBER		Identification code of the resource list
P_LIST_RETURN_STATUS	OUT	VARCHAR2(1)		Return status of the specific resource list

INIT_CREATE_RESOURCE_LIST

INIT_CREATE_RESOURCE_LIST is a Load–Execute–Fetch procedure used to set up the global data structures used by other Load–Execute–Fetch procedures.

INIT_UPDATE_MEMBERS

INIT_UPDATE_MEMBERS is a Load–Execute–Fetch procedure used to set up the global data structures used by other Load–Execute–Fetch procedures.

LOAD_MEMBERS

LOAD_MEMBERS is a Load–Execute–Fetch procedure used to load the resource list member global input structure.

Business Rules

- Calling modules can pass either P_RESOURCE_LIST_NAME or P_RESOURCE_LIST_ID.
- If the calling modules pass both P_RESOURCE_LIST_NAME and P_RESOURCE_LIST_ID, the API uses the latter.
- If the resource list is grouped, you must pass a valid resource group alias.
- The value for P_RESOURCE_ATTR_VALUE must correspond to the value for P_RESOURCE_TYPE. For example, person identification code for P_RESOURCE_ATTR_VALUE must be valid if P_RESOURCE_TYPE equals EMPLOYEE.
- If the calling module passes information to this API for both resource group and resource member parameters, the API first verifies that the resource group exists. If the resource group does not exist, the API creates the resource group and then creates the resource.
- If a given resource member already exists, this API does not return an error. Instead, it returns a successful return status and the resource list member identification code of the existing resource member.

  Note: Because you can store only one transaction attribute for a given resource, this API accepts only a single RESOURCE_ATTR_VALUE, which may hold PERSON_ID, JOB_ID, and so on.
- You can use the P_ENABLED_FLAG to enable or disable a resource member. If the parameter value is passed as NULL or something other than Y, the column value remains the same.

  Note: The alias must be unique within a resource group.

The following table shows the parameters for LOAD_MEMBERS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RESOURCE_GROUP_ALIAS	IN	VARCHAR2(30)	No	Alias of the resource group
P_RESOURCE_GROUP_NAME	IN	VARCHAR2(30)	No	Name of the resource group

Name	Usage	Type	Req?	Description
P_RESOURCE_TYPE_CODE	IN	VARCHAR2(30)	No	Type of the resource
P_RESOURCE_ATTR_VALUE	IN	VARCHAR2(80)	No	Attribute value of the resource
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	Alias of the resource member
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	Identification code of the resource list member
P_NEW_ALIAS	IN	VARCHAR2(30)	No	New alias of the resource member
P_SORT_ORDER	IN	NUMBER	No	Sort order of the resource member
P_ENABLED_FLAG	OUT	VARCHAR2(1)	No	Enabled flag of the resource member (default = 'Y')
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

LOAD_RESOURCE_LIST

LOAD_RESOURCE_LIST is a Load–Execute–Fetch procedure used to load the resource list global input structure.

Business Rules

- Valid values for P_GROUP_RESOURCE_TYPE are EXPENDITURE_CATEGORY, REVENUE_CATEGORY, ORGANIZATION, and NONE.
- The resource list name must be unique.
- If calling programs pass the P_MEMBER_TBL (optional), this API creates the relevant resource list member records.
- If your resource list is grouped, you must pass a valid resource group alias.
- The value for P_RESOURCE_ATTR_VALUE must correspond with the value for P_RESOURCE_TYPE. For example, P_RESOURCE_ATTR_VALUE must have a valid person identification code if P_RESOURCE_TYPE equals EMPLOYEE.
- If the value for GROUP_RESOURCE_TYPE is NONE, this API ignores resource group IN parameters.
- If you do not specify the resource group alias, the group resource type must be NONE.
- Calling modules can pass either P_RESOURCE_LIST_NAME or P_RESOURCE_LIST_ID.

- If the calling modules pass both P_RESOURCE_LIST_NAME and P_RESOURCE_LIST_ID, this API uses only the latter.
- If the resource list already contains active members, you cannot change GROUPED_BY_TYPE.
- You can change the following fields at any time:
  - RESOURCE_LIST_NAME
  - DESCRIPTION
  - START DATE
  - END DATE
- To update existing or add new resource list members, include the member records in MEMBER_TBL. If a resource list member already exists, you can update the following fields:
  - ALIAS. Specify P_NEW_ALIAS.
  - SORT_ORDER. Specify P_SORT_ORDER.
- You can use the value for P_ALIAS as the key to fetch the member record.

Note: The alias must be unique within a resource group.

The following table shows the parameters for LOAD_RESOURCE_LIST.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_RESOURCE_LIST_NAME	IN	VARCHAR2(30)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	Identification code of resource list
P_GROUP_RESOURCE_TYPE	IN	VARCHAR2(30)	No	Type of the resource group
P_DESCRIPTION	IN	VARCHAR2(80)	No	Description
P_START_DATE	IN	DATE	No	Start date of the resource list
P_END_DATE	IN	DATE	No	End date of the resource list
P_NEW_LIST_NAME	IN	VARCHAR2(30)	No	New name of the resource list
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

CHAPTER

4

Oracle Project Costing APIs

This chapter describes how to implement APIs that interface and assign assets from external systems.

Asset APIs

The asset APIs provide an open interface for external systems to insert, update, assign, and delete assets.

Asset API Views

The following table lists the views that provide parameter data for the asset APIs. For detailed description of the views, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

View	Description
PA_PROJECT_ASSET_TYPE_LOV_V	You can use this view to retrieve valid project asset types from Oracle Projects and display them in your external system.
PA_ASSET_BOOKS_LOV_V	You can use this view to retrieve valid asset books from Oracle Projects and display them in your external system.
PA_PARENT_ASSET_LOV_V	You can use this view to retrieve valid parent assets from Oracle Projects and display them in your external system.
PA_RET_TARGET_ASSET_LOV_V	You can use this view to retrieve valid retired target assets from Oracle Projects and display them in your external system.

Table 4 – 1 Asset API views (Page 1 of 1)

Asset API Procedures

The procedures discussed in this section are listed below. The procedures are located in the public API package PA_PROJECT_ASSETS_PUB.

- ADD_PROJECT_ASSET: page 4 – 4
- UPDATE_PROJECT_ASSET: page 4 – 6
- DELETE_PROJECT_ASSET: page 4 – 8
- ADD_ASSET_ASSIGNMENT: page 4 – 9

Asset API Procedure Definitions

This section contains detailed description of the asset APIs.

ADD_PROJECT_ASSET

This procedure adds a project asset to the specified project If the validations complete successfully, a new PA_PROJECT_ASSETS_ALL row is created.

The following table shows the parameters for ADD_PROJECT_ASSET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_ASSET_REFERENCE	IN	VARCHAR2(240)	Yes	The reference code that uniquely identifies the asset in the external system
P_PA_ASSET_NAME	IN	VARCHAR2(240)	Yes	The name that uniquely defines the asset in Oracle Projects
P_ASSET_NUMBER	IN	VARCHAR2(15)	Yes	Unique asset number
P_ASSET_DESCRIPTION	IN	VARCHAR2(80)	Yes	Asset description
P_PROJECT_ASSET_TYPE	IN	VARCHAR2(30)	Yes	Asset type
P_LOCATION_ID	IN	NUMBER	No	The identifier of the location to which the asset is assigned

Name	Usage	Type	Req?	Description
P_ASSIGNED_TO_PERSON_ID	IN	NUMBER	No	The identifier of the person to whom the asset is assigned
P_DATE_PLACED_IN_SERVICE	IN	DATE	No	Date placed in service of the asset
P_ASSET_CATEGORY_ID	IN	NUMBER	No	The identifier of the asset category to which the asset is assigned
P_BOOK_TYPE_CODE	IN	VARCHAR2(15)	No	The corporate book to which the asset is assigned
P_ASSET_UNITS	IN	NUMBER	No	The number of asset units
P_ESTIMATED_ASSET_UNITS	IN	NUMBER	No	The estimated number of asset units
P_ESTIMATED_COST	IN	NUMBER	No	The estimated cost
P_DEPRECIATE_FLAG	IN	VARCHAR2(1)	No	Indicator whether the asset should be depreciated in Oracle Assets
P_DEPRECIATE_EXPENSE_CCID	IN	NUMBER	No	The depreciation expense account for the asset
P_AMORTISE_FLAG	IN	VARCHAR2(1)	No	Indicator whether cost adjustments should be ammortised in Oracle Assets
P_ESTIMATED_IN_SERVICE_DATE	IN	DATE	No	The estimated date placed in service for the asset
P_ASEET_KEY_CCID	IN	NUMBER	No	Key flexfield code combination identifier for asset key flexfield
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Descriptive flexfield segment
P_PARENT_ASSET_ID	IN	NUMBER	No	The identifier of the parent asset
P_MANUFACTUREER_NAME	IN	VARCHAR2(30)	No	The name of the manufacturer of the asset
P_MODEL_NUMBER	IN	VARCHAR2(40)	No	The model number of the asset
P_SERIAL_NUMBER	IN	VARCHAR2(35)	No	The serial number of the asset
P_TAG_NUMBER	IN	VARCHAR2(15)	No	The tag number of the asset
P_RET_TARGET_ASSET_ID	IN	NUMBER	No	The identifier of the target asset
P_PA_PROJECT_ID_OUT	OUT	NUMBER(15)		API standard
P_PA_PROJECT_NUMBER_OUT	OUT	VARCHAR2(25)		API standard

Name	Usage	Type	Req?	Description
P_PA_PROJECT_ASSET_ID_OUT	OUT	NUMBER(15)		The reference code that uniquely identifies the asset within a project in Oracle Projects
P_PM_ASSET_REFERENCE_OUT	OUT	VARCHAR2(25)		The reference code that uniquely identifies the asset in the external system

UPDATE_PROJECT_ASSET

This procedure updates a project asset on the specified project. If the validations complete successfully, the PA_PROJECT_ASSETS_ALL row is updated with any new values specified.

The following table shows the parameters for UPDATE_ASSET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	YES	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER	Yes	API standard
P_MSG_DATA	OUT	VARCHAR2(2000)	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_ASSET_REFERENCE	IN	VARCHAR2(25)		The reference code that uniquely identifies the asset in the external system
P_PA_PROJECT_ASSET_ID	IN	NUMBER(15)		The reference code that uniquely identifies the asset within a project in Oracle Projects
P_PA_ASSET_NAME	IN	VARCHAR2(240)	Yes	The name that uniquely defines the asset in Oracle Projects
P_ASSET_NUMBER	IN	VARCHAR2(15)	Yes	Unique asset number
P_ASSET_DESCRIPTION	IN	VARCHAR2(80)	Yes	Asset description

Name	Usage	Type	Req?	Description
P_PROJECT_ASSET_TYPE	IN	VARCHAR2(30)	Yes	Asset type
P_LOCATION_ID	IN	NUMBER	No	The identifier of the location to which the asset is assigned
P_ASSIGNED_TO_PERSON_ID	IN	NUMBER	No	The identifier of the person to whom the asset is assigned
P_DATE_PLACED_IN_SERVICE	IN	DATE	No	Date placed in service of the asset
P_ASSET_CATEGORY_ID	IN	NUMBER	No	The identifier of the asset category to which the asset is assigned
P_BOOK_TYPE_CODE	IN	VARCHAR2(15)	No	The corporate book to which the asset is assigned
P_ASSET_UNITS	IN	NUMBER	No	The number of asset units
P_ESTIMATED_ASSET_UNITS	IN	NUMBER	No	The estimated number of asset units
P_ESTIMATED_COST	IN	NUMBER	No	The estimated cost
P_DEPRECIATE_FLAG	IN	VARCHAR2(1)	No	Indicator whether the asset should be depreciated in Oracle Assets
P_DEPRECIATE_EXPENSE_CCID	IN	NUMBER	No	The depreciation expense account for the asset
P_AMORTISE_FLAG	IN	VARCHAR2(1)	No	Indicator whether cost adjustments should be ammortised in Oracle Assets
P_ESTIMATED_IN_SERVICE_DATE	IN	DATE	No	The estimated date placed in service for the asset
P_ASEET_KEY_CCID	IN	NUMBER	No	Key flexfield code combination identifier for asset key flexfield
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Descriptive flexfield segment
P_PARENT_ASSET_ID	IN	NUMBER	No	The identifier of the parent asset
P_MANUFACTUREER_NAME	IN	VARCHAR2(30)	No	The name of the manufacturer of the asset
P_MODEL_NUMBER	IN	VARCHAR2(40)	No	The model number of the asset
P_SERIAL_NUMBER	IN	VARCHAR2(35)	No	The serial number of the asset
P_TAG_NUMBER	IN	VARCHAR2(15)	No	The tag number of the asset
P_RET_TARGET_ASSET_ID	IN	NUMBER	No	The identifier of the target asset
P_PA_PROJECT_ID_OUT	OUT	NUMBER(15)		API standard
P_PA_PROJECT_NUMBER_OUT	OUT	VARCHAR2(25)		API standard

Name	Usage	Type	Req?	Description
P_PA_PROJECT_ASSET_ID_OUT	OUT	NUMBER(15)		The reference code that uniquely identifies the asset within a project in Oracle Projects
P_PM_ASSET_REFERENCE_OUT	OUT	VARCHAR2(25)		The reference code that uniquely identifies the asset in the external system

DELETE_PROJECT_ASSET

This procedure deletes a project asset and any associated asset assignments from a project.

The following table shows the parameters for DELETE_PROJECT_ASSET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	No	The identifier of the external project management system from which the project was imported
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_ASSET_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the asset in the external system
P_PA_PROJECT_ASSET_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the asset within a project in Oracle Projects

ADD_ASSET_ASSIGNMENT

This procedure adds an asset assignment to the specified project. If the validations complete successfully, a PA_PROJECT_ASSET_ASSIGNMENTS row is created.

The following table shows the parameters for ADD_ASSET_ASSIGNMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER	Yes	API standard
P_MSG_DATA	OUT	VARCHAR2(2000)	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	Yes	See the TASK_IN_TBL_TYPE Datatype table on page 3 – 15 for a description of this field.
P_PA_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_ASSET_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the asset in the external system
P_PA_PROJECT_ASSET_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the asset within a project in Oracle Projects
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Descriptive flexfield segment
P_PA_TASK_ID_OUT	OUT	NUMBER(15)		The reference code that uniquely identifies the task within a project in Oracle Projects
P_PA_PROJECT_ASSET_ID_OUT	OUT	NUMBER(15)		The reference code that uniquely identifies the asset within a project in Oracle Projects

DELETE_ASSET_ASSIGNMENT

This procedure deletes an asset assignment from a project.

The following table shows the parameters for DELETE_ASSET_ASSIGNMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER	Yes	API standard
P_MSG_DATA	OUT	VARCHAR2(2000)	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	Yes	See the TASK_IN_TBL_TYPE Datatype table on page 3 – 15 for a description of this field.
P_PA_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_ASSET_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the asset in the external system
P_PA_PROJECT_ASSET_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the asset within a project in Oracle Projects

LOAD_PROJECT_ASSET

This procedure adds a project asset row to the global PL/SQL table G_ASSETS_IN_TBL. If the asset already exists on the project, the procedure calls the update_project_asset procedure.

The following table shows the parameters for LOAD_PROJECT_ASSET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_ASSET_REFERENCE	IN	VARCHAR2(240)	Yes	The reference code that uniquely identifies the asset in the external system
P_PA_ASSET_NAME	IN	VARCHAR2(240)	Yes	The name that uniquely defines the asset in Oracle Projects
P_ASSET_NUMBER	IN	VARCHAR2(15)	Yes	Unique asset number
P_ASSET_DESCRIPTION	IN	VARCHAR2(80)	Yes	Asset description
P_PROJECT_ASSET_TYPE	IN	VARCHAR2(30)	Yes	Asset type
P_LOCATION_ID	IN	NUMBER	No	The identifier of the location to which the asset is assigned
P_ASSIGNED_TO_PERSON_ID	IN	NUMBER	No	The identifier of the person to whom the asset is assigned
P_DATE_PLACED_IN_SERVICE	IN	DATE	No	Date placed in service of the asset
P_ASSET_CATEGORY_ID	IN	NUMBER	No	The identifier of the asset category to which the asset is assigned
P_BOOK_TYPE_CODE	IN	VARCHAR2(15)	No	The corporate book to which the asset is assigned
P_ASSET_UNITS	IN	NUMBER	No	The number of asset units
P_ESTIMATED_ASSET_UNITS	IN	NUMBER	No	The estimated number of asset units
P_ESTIMATED_COST	IN	NUMBER	No	The estimated cost
P_DEPRECIATE_FLAG	IN	VARCHAR2(1)	No	Indicator whether the asset should be depreciated in Oracle Assets
P_DEPRECIATE_EXPENSE_CCID	IN	NUMBER	No	The depreciation expense account for the asset
P_AMORTIZE_FLAG	IN	VARCHAR2(1)	No	Indicator whether cost adjustments should be ammortised in Oracle Assets
P_ESTIMATED_IN_SERVICE_DATE	IN	DATE	No	The estimated date placed in service for the asset
P_ASEET_KEY_CCID	IN	NUMBER	No	Key flexfield code combination identifier for asset key flexfield
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Descriptive flexfield segment
P_PARENT_ASSET_ID	IN	NUMBER	No	The identifier of the parent asset

Name	Usage	Type	Req?	Description
P_MANUFACTUREER_NAME	IN	VARCHAR2(30)	No	The name of the manufacturer of the asset
P_MODEL_NUMBER	IN	VARCHAR2(40)	No	The model number of the asset
P_SERIAL_NUMBER	IN	VARCHAR2(35)	No	The serial number of the asset
P_TAG_NUMBER	IN	VARCHAR2(15)	No	The tag number of the asset
P_RET_TARGET_ASSET_ID	IN	NUMBER	No	The identifier of the target asset

LOAD_ASSET_ASSIGNMENT

This procedure adds an asset assignment row to the global PL/SQL table G_ASSET_ASSIGNMENTS_IN_TBL. Rows in this table can then be added in mass to the current project by the execute_add_project_asset procedure, which calls the add_asset_assignment procedure for each row in the PL/SQL table.

The following table shows the parameters for LOAD_ASSET_ASSIGNMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	Yes	See the TASK_IN_TBL_TYPE Datatype table on page 3 – 15 for a description of this field.
P_PA_TASK_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_ASSET_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the asset in the external system
P_PA_PROJECT_ASSET_ID	IN	NUMBER(15)	No	The reference code that uniquely identifies the asset within a project in Oracle Projects
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Descriptive flexfield segment

EXECUTE_ADD_PROJECT_ASSET

This procedure is called from the create_project procedure. It processes project assets and project asset assignments sent to the procedure in PL/SQL table input parameters.

For each project asset row in the P_ASSETS_IN table, the procedure determines if the asset already exists. If it exists, the procedure calls the update_project_asset procedure for that row. Otherwise, it calls the add_project_asset procedure for that row.

For each project asset assignment row in the P_ASSET_ASSIGNMENTS_IN table, the procedure determines if the asset assignment already exists. If the assignment does not exist, the procedure calls the add_asset_assignment procedure for that row. If it does exist, the procedure does nothing.

The following table shows the parameters for EXECUTE_ADD_PROJECT_ASSET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_MSG_COUNT	OUT	NUMBER	Yes	API standard
P_MSG_DATA	OUT	VARCHAR2(2000)	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the project in the external system
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_ASSETS_IN	IN	ASSET_IN_TBL_TYPE	Yes	
P_ASSETS_OUT	OUT NOCOPY	ASSET_OUT_TBL_TYPE		

Name	Usage	Type	Req?	Description
P_ASSET_ASSIGNMENTS_IN	IN	ASSET_ASSIGNMENT_IN_TBL_TYPE	No	
P_ASSET_ASSIGNMENTS_OUT	OUT NOCOPY	ASSET_ASSIGNMENT_OUT_TBL_TYPE		

Cost Plus Application Programming Interface (API)

Oracle Projects provides a procedure you can use to call the Cost Plus Application Programming Interface. This procedure retrieves an amount based on your burden cost setup. You can specify the burden schedule, effective date, expenditure type, and organization to retrieve the burden cost amount based on the criteria you specify.

For example, you can use this procedure to derive the raw cost amount of a related transaction using a specific burden schedule of rates and the project organization as inputs.

> **Note:** Any amounts calculated using the API will not show up in cost plus detail views that display the burden cost breakdown. Also, if you update rates for the burden schedule, you must manually mark all items that are affected by the rate changes.

Procedure: Get Burden Amount

The cost plus application programming interface procedure is *pa_cost_plus.get_burden_amount*.

The following table lists the parameters that Oracle Projects provides for the pa_cost_plus.get_burden_amount procedure.

Parameter	Usage	Type	Description
burden_schedule_id	IN	NUMBER	The schedule id of the burden schedule used to calculate the burden amount.
effective_date	IN	DATE	The date used to find the burden schedule revision to calculate the burden amount.
expenditure_type	IN	VARCHAR2	The type of expenditure item used to find a cost base.
organization_id	IN	NUMBER	The id of the organization used to find a multiplier.

Table 4 – 2 (Page 1 of 2)

Parameter	Usage	Type	Description
raw_amount	IN	NUMBER	The raw amount for which the burden amount is calculated.
burden_amount	IN OUT	NUMBER	The calculated burden amount.
burden_sch_rev_id	IN OUT	NUMBER	The schedule revision id of the burden schedule used to calculate the burden amount.
compiled_set_id	IN OUT	NUMBER	The id of the active compiled set used to calculate the burden amount.
status	IN OUT	NUMBER	The processing status of the procedure.
stage	IN OUT	NUMBER	The exit stage of the procedure.

Table 4 – 2 (Page 2 of 2)

Error Handling

Use the status and stage parameters to help resolve error conditions should your procedure fail.

The status parameter indicates the processing status of your procedure as follows:

status = 0 The procedure executed successfully.

status < 0 An Oracle error occurred and the process did not complete.

Suggestion: Ensure that you are returning the status of the cost plus procedure to the procedure that you are calling the cost plus API from to help resolve error conditions.

status > 0 See stage parameter.

The stage parameter shows you where in the processing of the cost plus API the procedure failed. Use the stage parameter to resolve the specific problem that caused your procedure to fail. The following table lists these different stages and what they mean.

Stage	Meaning
100	Cannot find a revision for the given burden schedule and effective date
200	Cannot find the burden structure
300	Expenditure type is not in a cost base in the burden structure
400	There is no active compiled set for the given burden schedule and organization
500	There is no compiled multiplier for the given qualification

Table 4 – 3 (Page 1 of 1)

See Also

Labor Transaction Extensions: page 9 – 22

Example of Using the Cost Plus API

This section gives an example of how to use the API to calculate the burden amount according to a specific business requirement.

The business requirement is to determine the burden amount based on the following criteria.

- Burden Schedule: CP burden schedule (burden schedule ID: 60)
- Effective Date: 03–MAR–94
- Expenditure Type: Professional
- Organization: Data Systems (Organization ID: 18)
- Raw Amount: 1,000

You would use the following PL/SQL procedure to obtain the burden amount for this business requirement using the cost plus API.

```
pa_cost_plus.get_burden_amount(60,
                        '03-MAR-94',
                        'Professional',
```

```
                        18,
                        1000,
                        burden_amount,
                        burden_sch_rev_id,
                        compiled_set_id,
                        status,
                        stage);
if (status = 0) then
   -- use the calculated burden_amount to implement your
   -- business requirement
end if;
```

CHAPTER

5

Oracle Project Billing APIs

This chapter describes how to implement APIs for:

- Agreements and funding
- Events

Agreement and Funding APIs

The agreement and funding APIs provide an open interface for external systems to insert, update, and delete agreements, as well as allocate funds from one agreement to any number of projects or top–level tasks.

Security for Agreement and Funding APIs

Actions performed using the APIs are subject to data level security (Control Actions). However, no function security is enforced. To maintain the same level of security as Oracle Projects, the APIs can only be executed through Oracle Applications. This enables you to log in to the database, choose a valid responsibility, and only access the APIs that the responsibility allows.

These APIs provide the ability to copy components from the agreements and funding form to create and maintain agreements and fundings.

Control Actions

The following new Control Actions have been added for Agreement / Funding API functionality:

- Update Agreement
- Delete Agreement
- Add Funding
- Update Funding
- Delete Funding

For more information on the control actions, see Control Actions Window, *Oracle Projects User Guide*.

Agreement and Funding API Views

The following table lists the views that provide parameter data for the agreement and funding APIs. For detailed description of the views, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

View	Description
PA_AGREEMENT_TYPE_LOV_V	Retrieves valid agreement types
PA_TERMS_LOV_V	Retrieves customer terms
PA_OWNED_BY_LOV_V	Retrieves valid employees
PA_CUSTOMERS_LOV_V	Retrieves valid customer names and numbers

Table 5 – 1 Agreement API Views (Page 1 of 1)

Agreement and Funding API Procedures

The procedures discussed in this section are listed below. The procedures are located in the public API package PA_AGREEMENT_PUB.

Agreement and Funding API Procedure Definitions

This section contains description of the agreement and funding APIs, including business rules and parameters.

CREATE_AGREEMENT

This API creates an agreement with associated funds.

> **Note:** To use this API you must have a database environment that is capable of supporting the PL/SQL table and a user defined record (for example, Oracle Server 7.3 and PL/SQL 2.3). Otherwise,use the Load–Execute–Fetch APIs supplied in the pa_agreement_pub_ package.

Business Rules

List of values

- Customer number
- Agreement type
- Agreement number
- Term name
- Revenue limit
- Valid Employee

The following table shows the parameters for CREATE_AGREEMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.

Name	Usage	Type	Req?	Description
P_AGREEMENT_IN_REC	IN	AGREEMENT_REC_IN_TYPE	Yes	The reference code that uniquely identifies the agreement input record in Oracle Projects.
P_AGREEMENT_OUT_REC	OUT	AGREEMENT_REC_OUT_TYPE		The reference code that uniquely identifies the agreement output record in Oracle Projects.
P_FUNDING_IN_TBL	IN	FUNDING_IN_TBL_TYPE	No	The reference code that uniquely identifies the funding input record in Oracle projects.
P_FUNDING_OUT_TBL	OUT	FUNDING_OUT_TBL_TYPE		The reference code that uniquely identifies the funding output record in Oracle Projects.

DELETE_AGREEMENT

This API deletes an agreement and associated funds.

Business Rules

- If the funding is baselined, the agreement cannot be deleted.
- Check accrued or billed amount:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for DELETE_AGREEMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.

Name	Usage	Type	Req?	Description
P_PM_AGREEMENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the agreement in the external system.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.

UPDATE_AGREEMENT

This API updates an agreement and associated funds.

Business Rules

- If there is at least one summary project funding that exists where the sum of the baselined amount and total unbaselined amount is less then the revenue accrued or billed amount, the API does not allow the revenue or invoice limit to be changed.
- The agreement amount cannot be be less than the sum of the total baselined amount and unbaselined amount.
- The customer cannot be changed if there is one fund for the agreement.
- List of Values
  - Customer number
  - Agreement type
  - Agreement number
  - Term name
  - Revenue limit
  - Valid employee

The following table shows the parameters for UPDATE_AGREEMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	YES	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER	Yes	API standard

Name	Usage	Type	Req?	Description
P_MSG_DATA	OUT	VARCHAR2(2000)	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.
P_AGREEMENT_IN_REC	IN	AGREEMENT_REC_IN_TYPE	Yes	The reference code that uniquely identifies the agreement input record in Oracle Projects.
P_AGREEMENT_OUT_REC	OUT	AGREEMENT_REC_OUT_TYPE		The reference code that uniquely identifies the agreement output record in Oracle Projects.
P_FUNDING_IN_TBL	IN	FUNDING_IN_TBL_TYPE	No	The reference code that uniquely identifies the funding input record in Oracle projects.
P_FUNDING_OUT_TBL	OUT	FUNDING_OUT_TBL_TYPE		The reference code that uniquely identifies the funding output record in Oracle Projects.

CREATE_BASELINE_BUDGET

The API procedure PA_AGREEMENT_PUB.CREATE_BASELINE_BUDGET creates and baselines an approved revenue budget and baselines the funding for a project. This procedure calls the PA_BUDGET_PUB.CREATE_DRAFT_BUDGET procedure to create a budget and the PA_BUDGET.BASELINE_BUDGET procedure to baseline the budget.

Business Rules:

- Baseline Funding without Budget must be enabled for the project. The functionality can be enabled for a project in the Revenue and Billing Information window.
- If funding for the project is at the project level, the procedure creates an approved revenue budget that uses the system–defined budget entry method Project Level Baseline. This budget entry method budgets at the project level and does not use a resource list.

- If funding for the project is at the top task level, the procedure creates an approved revenue budget that uses the system–defined budget entry method Task Level Baseline. This budget entry method budgets at the top task level and does not use a resource list.
- The currency of the budget is the project functional currency.
- If descriptive flexfields are defined for a budget, you can pass them in as parameters.
- All the business rules associated with the Budget APIs are enforced.

The following shows the parameters for the PA_AGREEMENT_PUB.CREATE_BASELINE_BUDGET

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2	No	API standard
P_INIT_MSG_LIST	IN	VARCHAR2	No	API standard
P_MSG_COUNT	OUT	NUMBER	Yes	API standard
P_MSG_DATA	OUT	VARCHAR2	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2	Yes	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2	Yes	The product code of the supplier of the external system.
P_PM_BUDGET_REFERENCE	IN	VARCHAR2	No	The reference code that uniquely identifies the budget in the external system.
P_PA_PROJECT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the project in Oracle Projects.
P_PM_PROJECT_REFERENCE	IN	VARCHAR2	No	The reference code that uniquely identifies a project in the external system.
P_CHANGE_REASON_CODE	IN	VARCHAR2	No	The reference code that identifies the change reason
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2	No	Budget descriptive flexfield

ADD_FUNDING

This API adds funding to an agreement.

Business Rules

- If the project is funded by multiple customers, funding cannot be done at the task level.
- If the project is funded by one customer, multiple agreements generate an error message.
- If the Project Type is not Contract, the fund amount must be zero.
- If the funding is baselined, the funding amount cannot be updated.
- If the project's invoice processing currency is defined as funding currency, the project cannot be funded by more than one currency.
- Check funding level: If there is an existing Project Level Funding, there cannot also be a Top Task Level Funding. A project can only have one funding level.
- Check accrued or billed amount:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for ADD_FUNDING.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(35)	Yes	The reference code that uniquely identifies the funding in the external system.
P_FUNDING_ID	IN OUT	NUMBER(15)	Yes	The reference code that uniquely identifies the funding in Oracle Projects.

Name	Usage	Type	Req?	Description
P_PA_PROJECT_ID	IN	NUMBER(15)	Yes	The reference code that uniquely identifies the project in Oracle Projects.
P_PA_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.
P_ALLOCATED_AMOUNT	IN	NUMBER	Yes	The reference code that uniquely identifies the allocated funding amount within a project in Oracle Projects.
P_DATE_ALLOCATED	IN	DATE	No	The reference code that uniquely identifies the date allocated within a project in Oracle Projects.
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_PROJECT_RATE_DATE	IN	DATE	No	Funding currency to project currency exchange rate date
P_PROJECT_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project currency exchange rate
P_PROJFUNC_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project functional currency exchange rate type
P_PROJFUNC_RATE_DATE	IN	DATE	No	Funding currency to project functional currency exchange rate date
P_PROJFUNC_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project functional currency exchange rate
P_FUNDING_CATEGORY	IN	VARCHAR2	No	The identifier of the funding category
P_DESC_FLEX_NAME	IN	VARCHAR2	No	Descriptive flexfield name
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Descriptive flexfield category
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfield category
P_FUNDING_ID_OUT	OUT	NUMBER		The reference code that uniquely identifies the funding within a project in Oracle Projects.

DELETE_FUNDING

This API deletes a fund from an agreement.

Business Rules

- If the funding is baselined, the agreement cannot be deleted.
- Check accrued or billed amount:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for DELETE_FUNDING.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(35)	Yes	The reference code that uniquely identifies the supplier funding in the external system.
P_FUNDING_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the funding within a project in Oracle Projects.
P_CHECK_Y_N	IN	VARCHAR2(1)	No	Flag indicating to check whether the funding line can be deleted

UPDATE_FUNDING

This API updates a fund for an agreement.

Business Rules

- If the project is funded by multiple customers, task level funding is not allowed.
- If the project is funded by one customer, multiple agreements generate an error message.
- If the Project Type is not Contract, the fund amount must be zero.
- If the funding is baselined, the funding amount cannot be updated.
- If the project's invoice processing currency is defined as funding currency, the project cannot be funded by more than one currency.
- Check funding level: If there is an existing Project Level Funding, there cannot also be a Top Task Level Funding. A project can only have one funding level.
- Check accrued or billed amount:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for UPDATE_FUNDING.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(35)	Yes	The reference code that uniquely identifies the supplier funding in the external system.
P_FUNDING_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the funding within a project in Oracle Projects.

Name	Usage	Type	Req?	Description
P_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies a project in Oracle Projects.
P_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies a task within a project in Oracle Projects.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.
P_ALLOCATED_AMOUNT	IN	NUMBER	No	The reference code that uniquely identifies the amount of funding allocated within a project in Oracle Projects.
P_DATE_ALLOCATED	IN	DATE	No	The reference code that uniquely identifies the allocated date within a project in Oracle Projects.
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_PROJECT_RATE_DATE	IN	DATE	No	Funding currency to project currency exchange rate date
P_PROJECT_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project currency exchange rate
P_PROJFUNC_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project functional currency exchange rate type
P_PROJFUNC_RATE_DATE	IN	DATE	No	Funding currency to project functional currency exchange rate date
P_PROJFUNC_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project functional currency exchange rate
P_FUNDING_CATEGORY	IN	VARCHAR2	No	The identifier of the funding category
P_DESCFLEX_NAME	IN	DATE	No	Descriptive flexfield name
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Descriptive flexfield category
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfield attribute
P_FUNDING_ID_OUT	OUT	NUMBER		The reference code that uniquely identifies the funding (outflows) within a project in Oracle Projects.

INIT_AGREEMENT

This API sets the global tables used by the Load–Execute–Fetch procedures that create a new agreement or update an existing agreement.

Parameters: None

LOAD_AGREEMENT

This API loads an agreement to a PL/SQL record.

The following table shows the parameters for LOAD_AGREEMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_AGREEMENT_ REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the agreement in the external system.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement within a project in Oracle Projects.
P_CUSTOMER_ID	IN	NUMBER	Yes	The identification code of the project's customer in Oracle Projects.
P_CUSTOMER_NAME	IN	VARCHAR2(50)	Yes	The identification name of the project's customer in Oracle Projects.
P_CUSTOMER_NUM	IN	VARCHAR2(30)	Yes	The identification number of the project's customer in Oracle Projects.
P_AGREEMENT_NUM	IN	VARCHAR2(20)	Yes	The reference code that uniquely identifies a agreement number within a project in Oracle Projects.
P_AGREEMENT_TYPE	IN	VARCHAR2(30)	Yes	The reference code that uniquely identifies a agreement type within a project in Oracle Projects.
P_AMOUNT	IN	NUMBER	Yes	The reference code that uniquely identifies the amount of the agreement within a project in Oracle Projects.
P_TERM_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the terms of the agreement within a project in Oracle Projects.

Name	Usage	Type	Req?	Description
P_TERM_NAME	IN	VARCHAR2(15)	Yes	The name that uniquely identifies the term of the agreement within a project in Oracle Projects.
P_REVENUE_LIMIT_FLAG	IN	VARCHAR2(1)	No	Indicates whether or not the revenue limit has been exceeded.
P_EXPIRATION_DATE	IN	DATE	No	Indicates the expiration date of the agreement within a project in Oracle Projects.
P_DESCRIPTION	IN	VARCHAR2(240)	No	Description of the agreement within a project in Oracle Projects.
P_OWNED_BY_PERSON_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the person who owns the agreement within a project in Oracle Projects.
P_OWNED_BY_PERSON_NAME	IN	VARCHAR2(240)	Yes	The name that uniquely identifies the person who owns the agreement within a project in Oracle Projects.
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_PROJECT_RATE_DATE	IN	DATE	No	Funding currency to project currency exchange rate date
P_PROJECT_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project currency exchange rate
P_PROJFUNC_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project functional currency exchange rate type
P_PROJFUNC_RATE_DATE	IN	DATE	No	Funding currency to project functional currency exchange rate date
P_PROJFUNC_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project functional currency exchange rate
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Descriptive flexfield category
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfield attribute
P_TEMPLATE_FLAG	IN	VARCHAR2(1)	No	Indicates whether or not the project is a template.
P_DESC_FLEX_NAME	IN	VARCHAR2(40)	No	Descriptive flexfield name

LOAD_FUNDING

This API loads funding to a PL/SQL table.

The following table shows the parameters for LOAD_FUNDING.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(35)	Yes	The reference code that uniquely identifies the funding in external system.
P_FUNDING_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the funding in Oracle Projects.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.
P_PROJECT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the project in Oracle Projects.
P_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects.
P_ALLOCATED_AMOUNT	IN	NUMBER	Yes	The reference code that uniquely identifies the amount of funding allocated within a project in Oracle Projects.
P_DATE_ALLOCATED	IN	DATE	No	The reference code that uniquely identifies the date funding was allocated within a project in Oracle Projects.
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_PROJECT_RATE_DATE	IN	DATE	No	Funding currency to project currency exchange rate date
P_PROJECT_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project currency exchange rate
P_PROJFUNC_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project functional currency exchange rate type
P_PROJFUNC_RATE_DATE	IN	DATE	No	Funding currency to project functional currency exchange rate date
P_PROJFUNC_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project functional currency exchange rate
P_FUNDING_CATEGORY	IN	VARCHAR2	No	The identifier of the funding category
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Descriptive flexfield category
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfield attribute

EXECUTE_CREATE_AGREEMENT

This API creates an agreement with the funding using the data stored in the global tables during the Load phase.

Business Rules

List of values

- Customer number
- Agreement type
- Agreement number
- Term name
- Revenue limit
- Valid Employee

The following table shows the parameters for EXECUTE_CREATE_AGREEMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.
P_AGREEMENT_ID_OUT	OUT	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.
P_CUSTOMER_ID_OUT	OUT	NUMBER	Yes	The reference code that uniquely identifies the customer in Oracle Projects.

EXECUTE_UPDATE_AGREEMENT

This API updates an agreement with the funding using the data stored in the global tables during the Load phase.

Business Rules

- If there is at least one summary project funding that exists where the sum of the baselined amount and total unbaselined amount is less then the revenue accrued or billed amount, the API does not allow the revenue or invoice limit to be changed.
- The agreement amount cannot be be less than the sum of the total baselined amount and unbaselined amount.
- The customer cannot be changed if there is one fund for the agreement.
- List of Values
  - Customer number
  - Agreement type
  - Agreement number
  - Term name
  - Revenue limit
  - Valid employee

The following table shows the parameters for EXECUTE_UPDATE_AGREEMENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default ='F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default ='F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.

FETCH_FUNDING

This API gets the return_status that was returned during creation of funds and stored in a global PL/SQL table.

The following table shows the parameters for FETCH_FUNDING.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	YES	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_FUNDING_INDEX	IN	NUMBER	Yes	Pointer to specific funding amount
P_FUNDING_ID	OUT	NUMBER		The reference code that uniquely identifies the funding in Oracle Projects.
P_PM_FUNDING_REFERENCE	OUT	VARCHAR2(35)		The reference code that uniquely identifies the funding in the external system.

CLEAR_AGREEMENT

This API clears the globals that were set up during initialization.

Parameters: None

CHECK_DELETE_AGREEMENT_OK

This API checks whether an agreement can be deleted.

Business Rules

- If the funding is baselined, the agreement cannot be deleted.
- Check accrued or billed amount:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for CHECK_DELETE_AGREEMENT_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard

Name	Usage	Type	Req?	Description
P_RETURN_STATUS	OUT	VARCHAR2(1)		APi standard
P_PM_AGREEMENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the agreement in the external system.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.
P_DEL_AGREE_OK_FLAG	OUT	VARCHAR2(1)		Boolean flag for deleting agreement

CHECK_ADD_FUNDING_OK

This API checks whether a fund can be added.

Business Rules

- If the project is funded by multiple customers, task level funding is not allowed.
- If the project is funded by one customer, multiple agreements generate an error message.
- If the project's invoice processing currency is defined as funding currency, the project cannot be funded by more than one currency.
- If the project type is not Contract, the fund amount must be zero.
- If the funding is baselined, the funding amount cannot be updated.
- The funding level must be valid: If there is an existing Project Level Funding, there cannot also be a Top Task Level Funding. A project can only have one funding level.
- The accrued / billed amount must be valid:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for CHECK_ADD_FUNDING_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_AGREEMENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the agreement in the external system.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle Projects.
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the funding in the external system.
P_TASK_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the task within the project in Oracle PRojects.
P_PROJECT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the project in Oracle Projects.
P_ADD_FUNDING_OK_FLAG	OUT	VARCHAR2(1)		Boolean flag for adding funding
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_PROJECT_RATE_DATE	IN	DATE	No	Funding currency to project currency exchange rate date
P_PROJECT_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project currency exchange rate
P_PROJFUNC_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project functional currency exchange rate type
P_PROJFUNC_RATE_DATE	IN	DATE	No	Funding currency to project functional currency exchange rate date
P_PROJFUNC_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project functional currency exchange rate
P_FUNDING_AMT	IN	NUMBER	No	Allocated funding amount

CHECK_DELETE_FUNDING_OK

This API checks whether a fund can be deleted.

Business Rules

- If the funding is baselined, the agreement cannot be deleted.
- Check accrued or billed amount:

  agreement amount >= total funding amount >=0

  AND

  total funding amount >= amount accrued or billed

The following table shows the parameters for CHECK_DELETE_FUNDING_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(35)	Yes	The reference code that uniquely identifies the funding in the external system.
P_FUNDING_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the funding in Oracle Projects.
P_DEL_FUNDING_OK_FLAG	OUT	VARCHAR2(1)		Boolean flag for deleting funding

CHECK_UPDATE_FUNDING_OK

This API checks whether a fund can be added.

Business Rules

- If the project is funded by multiple customers, task level funding is not allowed.
- If the project type is not Contract, the fund amount must be zero.
- If the funding is baselined, the funding amount cannot be updated.
- If the project's invoice processing currency is defined as funding currency, the project cannot be funded by more than one currency.

- Funding level checks
  - If there is no task ID , there can be no task level funding.
  - If there is a task ID, there can be no project level funding.
- Check accrued / billed amount
  - agreement amount >= total funding amount >=0 AND
  - total funding amount >= amount accrued or billed

The following table shows the parameters for CHECK_UPDATE_FUNDING_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported.
P_PM_FUNDING_REFERENCE	IN	VARCHAR2(35)	Yes	The reference code that uniquely identifies the funding in the external system.
P_FUNDING_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the funding in Oracle Projects.
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the project in the external system.
P_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects.
P_PM_TASK_REFERENCE	IN	VARCHAR2(25)	No	The reference code that uniquely identifies the task in the external system.
P_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects.
P_PM_AGREEMENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the agreement in the external system.
P_AGREEMENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the agreement in Oracle projects.

Name	Usage	Type	Req?	Description
P_ALLOCATED_AMOUNT	IN	NUMBER	No	The reference code that uniquely identifies the amount of funding allocated within a project in Oracle Projects.
P_DATE_ALLOCATED	IN	DATE	No	The reference code that uniquely identifies the date funding was allocated within a project in Oracle Projects.
P_FUNDING_CATEGORY	IN	VARCHAR2	No	The identifier of the funding category
P_DESC_FLEX_NAME	IN	VARCHAR2(40)	No	Descriptive flexfield name
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Descriptive flexfield category
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfield attribute
P_UPDATE_FUNDING_OK_FLAG	OUT	VARCHAR2(1)		Boolean flag for deleting funding
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_OWNING_ORGANIZATION_ID	IN	NUMBER	No	Unique identifier of the owning organization
P_AGREEMENT_CURRENCY_CODE	IN	VARCHAR2(15)	No	Funding currency code for the agreement
P_INVOICE_LIMIT_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether invoices for projects funded by this agreement can exceed the allocated funding amount

Using Agreement and Funding APIs

The following example describes how to create an interface between Oracle Projects and the agreement and funding information entered in your system. Depending on your company's business needs, your implementation of the project APIs may be more or less complex than the scenario shown here. As you work through the example, you may want to refer to information elsewhere in the manual.

- For a detailed description of agreement and funding APIs, see Agreement and Funding APIs: page 5 – 2.
- Most of the Oracle Projects APIs use a standard set of input and output parameters. For a description of these parameters, see Standard API Parameters: page 2 – 19.
- For an example of PL / SQL code for creating a project without using composite datatypes, see Creating a Project Using the Load–Execute–Fetch APIs: page 3 – 76.

Step 1 Connect to an Oracle database

To ensure that proper security is enforced while accessing Oracle Projects data, follow the steps in Security Requirements: page 2 – 9.

Step 2 Collect agreement information

Collect the following information to create an agreement in Oracle Projects:

- Agreement reference– Unique identifier of the Agreement.
- Customer– Valid customer in Oracle Projects.
- Agreement Type– Valid agreement type in Oracle Projects.
- Agreement Terms– Valid agreement terms in Oracle Projects.
- Owner of the agreement– Valid employee in Oracle Projects.

You can also use the following views to retrieve the list of values for collecting agreement information:

- PA_AGREEMENT_TYPE_LOV_V
- PA_TERMS_LOV_V
- PA_OWNED_BY_LOV_V
- PA_CUSTOMERS_LOV_V

Step 3 Interface agreement information to the server

Not all tools can call the APIs that use composite datatypes. Tools that do not support composite datatypes must call the supplementary Load –Execute –Fetch APIs. The Load–Execute–Fetch APIs include procedures to initialize, load, execute, fetch, and clear data.

Use these APIs only if you use a tool that does not support composite data type parameters. If the tool (for example, Oracle PL/SQL Version 2.3 or higher) supports composite data type parameters, you can call the CREATE_AGREEMENT and ADD_FUNDING APIs directly. Following is the flow of the Load–Execute–Fetch Agreement and Funding procedures:

- Initialize Agreement (INIT_AGREEMENT)
- Load Agreement (LOAD_AGREEMENT)
- Load Funding (LOAD_FUNCING)
- Execute Create Agreement (EXECUTE_CREATE_AGREEMENT)
- Fetch Funding (Fetch Funding)
- Clear Agreement (CLEAR_AGREEMENT)

In the example above, INIT_AGREEMENT resets the server–side global PL/SQL tables that temporarily store the Agreement and Funding data. Once you set up these tables, you can use LOAD_AGREEMENT to move the Agreement data to the Oracle Projects database.

When you create a new agreement, this procedure must also pass parameters: P_PM_AGREEMENT_REFERENCE the unique reference code that identifies the agreement in the external system.

Step 4 Interface funding information to the server

After you interface the agreement–related data to the server, call LOAD_FUNDING to interface with the funding–related data to the server–side global PL/SQL tables. Call LOAD_ FUNDING once for each funding in the agreement.

Attention: Each funding must specify at least the following information:

- Funding Reference (P_PM_FUNDING_REFERENCE): The unique reference code that identifies the funding in the external system.
- Agreement ID (P_AGREEMENT_ID): The identifier of the agreement for which the funding needs to be created.

- Project ID (P_PROJECT_ID): The identifier of the Project for which the funding needs to be created.
- Task ID (P_TASK_ID): For task–level funding, the identifier of the task for which the funding needs to be created.

Step 5 Start the server–side process

Once the Load procedures have successfully moved the agreement and funding data to the Oracle Projects global PL / SQL tables, call up the procedure EXECUTE_CREATE_AGREMENT to process the agreement and funding data that you interfaced to the global PL / SQL tables. In addition to the standard input and output parameters, this Execute procedure requires the following parameters:

Input parameters

- P_PM_PRODUCT_CODE – The identification code of the product exporting the agreement . For information about setting up your product (external system) as a source, refer to Setting Up Your Product in Oracle Projects.

Output parameters

- P_AGREEMENT_ID – The unique Oracle Projects identification code for the new Agreement. .
- P_CUSTOMER_ID – The unique Oracle Projects customer id with which the agreement was created.

Step 6 Get return values for fundings

After the Load and Execute procedures create your agreement and funding in Oracle Projects, use FETCH_FUNDING to return each unique funding identification code from Oracle Projects.

The key input parameter for this procedure is P_FUNDING_INDEX, which points to a single funding,and the output parameters are P_FUNDING_ID and P_PM_FUNDING_REFERENCE.

To call the procedure for each funding, you can write a simple program to call FETCH_FUNDING in a loop with P_FUNDING_INDEX as the stepping variable (1 through the total number of funding). The output parameter P_RETURN_STATUS indicates whether the API handled the specific funding successfully (S). If the parameter returns E or U, the funding caused an error, and you must stop the Fetch procedure to retrieve the related error message. Fetch APIs do not return error message data. Instead, use GET_MESSAGES to retrieve the error text, as described in the next step.

Step 7 Retrieve error messages

Every Oracle Projects API includes two standard output parameters:

- P_RETURN_STATUS – indicates whether the API was executed successfully
- P_MSG_COUNT shows the number of errors detected during the execution of the API

If the API detects one error, the API returns the error message text. If the API detects multiple errors, use GET_MESSAGES to retrieve the error messages. See GET_MESSAGES: page 2 – 24.

Step 8 Finish the Load–Execute–Fetch process

After executing the Fetch procedures and retrieving any error messages, finish the Load–Execute–Fetch process by calling the API CLEAR_AGREEMENT and either save or roll back your changes to the database.

Creating an Agreement Using Load–Execute–Fetch APIs

The following sample PL/SQL code is a script that creates an agreement using the Load–Execute–Fetch APIs. The Load–Execute–Fetch APIs use parameters with standard datatypes (VARCHAR2, NUMBER, and DATE). These APIs do not use composite datatypes.

To create agreements using tools or products that support composite datatypes, see Creating an Agreement Using a Composite Datatype API: page 5 – 37.

```
DECLARE
  --API standard parameters

l_api_version_number                NUMBER :=1.0;
l_commit                            VARCHAR2(1):='T';
l_return_status                     VARCHAR2(1)
l_init_msg_list                     VARCHAR2(1)
l_msg_count                         NUMBER;
l_msg_data                          VARCHAR2(2000);
l_data                              VARCHAR2(2000);
l_msg_entity                        VARCHAR2(100);
l_msg_entity_index                  NUMBER;
l_msg_index                         NUMBER;
l_msg_index_out                     NUMBER;
l_encoded                           VARCHAR2(1)
l_agreement_id_out                  NUMBER;
l_customer_id_out             NUMBER;
l_funding_id                        NUMBER;

  --Oracle agreement specific variable

l_pm_product_code             VARCHAR2(25);
l_agreement_in_rec                  pa_agreement_pub.Agreem
ent_Rec_In_Type;
l_agreement_out_rec                 pa_agreement_pub.Agreem
ent_Rec_Out_Type;

..--Oracle funding specific parameters
l_funding_type                      pa_agreement_pub.fundin
g_rec_in_type;
l_funding_in-tbl                    pa_agreement_pub.fundin
g_in_tbl_type;
```

```
l_funding_out_tbl             pa_agreement_pub.funding_out_t
bl_type;

  --Local agreement parameters

l_early_start_date                    DATE;
l_pm_agreement_reference              VARCHAR2(25);
l_agreement_id                        NUMBER;
l_customer_id                         NUMBER;
l_customer_name                       VARCHAR2(25);
l_customer_num                        VARCHAR2(25);
l_agreement_num                       VARCHAR2(25);
l_agreement_type                      VARCHAR2(25);
l_amount                              NUMBER;
l_term_id                             NUMBER;
l_term_name                           VARCHAR2(25);
l_revenue_limit_flag                  VARCHAR2(25);
l_expiration_date             DATE;
l_description                         VARCHAR2(25);
l_owned_by_person_id                  NUMBER;
l_owned_by_person_name                VARCHAR2(25);
l_attribute_category                  VARCHAR2(25);
l_attribute1                          VARCHAR2(25);
l_attribute2                          VARCHAR2(25);
l_attribute3                          VARCHAR2(25);
l_attribute4                          VARCHAR2(25);
l_attribute5                          VARCHAR2(25);
l_attribute6                          VARCHAR2(25);
l_attribute7                          VARCHAR2(25);
l_attribute8                          VARCHAR2(25);
l_attribute9                          VARCHAR2(25);
l_attribute10                         VARCHAR2(25);
l_template_flag                       VARCHAR2(25);

..---local funding variables
l_pm_funding_reference                VARCHAR2(25);
l_funding_rec                         pa_agreement_pub.fundin
g_rec_in_type;
l_funding_in                          pa_agreement_pub.fundin
g_in_tbl_type;

  ---loop variables
a                                     NUMBER:=0;
API_ERROR                             EXCEPTION;
```

```
BEGIN

  --- PRODUCT RELATED DATA
l_pm_product_code             :='MSPROJECT';

  --- AGREEMENT RELATED DATA
l_pm_agreement-reference             :='amg06';
l_agreement_id                       :=Null;
l_customer_id                        :=1004;
l_customer_name                      :='Universal
Packaging';
l_customer_num                       :='1004';
l_agreement_num                      :='amg06';
l_agreement_type                     :='Service Agreement';
l_amount                              := 2000;
l_term_id                             := 4;
l_term_name                           := Null;
l_revenue_limit_flag                  := N;
l_expiration_date              := Null;
l_description                        := Null;
l_owned_by_person_id                 := 53;
l_owned-by_person_name               :=Null;
l_attribute_category                 :=Null;
l_attribute1                         :=Null;
l_attribute2                         :=Null;
l_attribute3                         :=Null;
l_attribute4                         :=Null;
l_attribute5                         :=Null;
l_attribute6                         :=Null;
l_attribute7                         :=Null;
l_attribute8                         :=Null;
l_attribute9                         :=Null;
l_attribute10                        :=Null;
l_template_flag                      :=N;

..---FUNDING RELATED DATA
a:= 1
l_funding_rec.pm_funding_reference   := 'amg06fun';
l_funding_rec.project_funding_id     = Null;
l_funding_rec.agreement_id           := Null;
l_funding_rec.project_id             := 15353;
l_funding_rec.task_id                := Null;
l_funding_rec.allocated_amount       := 1000;
```

```
l_funding_rec.date_allocated         := '01-JAN-2000';
l_funding_rec.attribute_category     := Null;
l_funding_rec.attribute1             := Null;
l_funding_rec.attribute2             := Null;
l_funding_rec.attribute3             := Null;
l_funding_rec.attribute4             := Null;
l_funding_rec.attribute5             := Null;
l_funding_rec.attribute6             := Null;
l_funding_rec.attribute7             := Null;
l_funding_rec.attribute8             := Null;
l_funding_rec.attribute9             := Null;
l_funding_rec.attribute10            := Null;


-- LOOP CONSTRUCT
 l_funding_in(a):= l_funding_rec;

  a:= 2;
l_funding_rec.pm_funding_reference   := 'C1004';
l_funding_rec.project_funding_id     :=Null;
l_funding_rec.agreement_id           := Null;
l_funding_rec.project_id             := 1404;
l_funding_rec.task_id                := Null;
l_funding_rec.allocated_amount       := 1000;
l_funding_rec.date_allocated         := '01-JAN-2000';
l_funding_rec.attribute_category     := Null;
l_funding_rec.attribute1             := Null;
l_funding_rec.attribute2             := Null;
l_funding_rec.attribute3             := Null;
l_funding_rec.attribute4             := Null;
l_funding_rec.attribute5             := Null;
l_funding_rec.attribute6             := Null;
l_funding_rec.attribute7             := Null;
l_funding_rec.attribute8             := Null;
l_funding_rec.attribute9                  := Null;
l_funding_rec.attribute10            := Null;

-- LOOP CONSTRUCT
    l_funding_in(a) := l_funding_rec;
-----------------------------------------------
--INIT_CREATE_AGREEMENT
pa_agreement_pub.init_agreement;
-----------------------------------------------
--LOAD AGREEMENT
```

```
pa_agreement_pub.load_agreement
    (p_api_version_number            => l_api_version_number
    ,p_init_msg_list                 => l_init_msg_list
    ,p_return_status                 => l_return_status
    ,p_pm_agreement_reference        =>
l_pm_agreement_reference
    ,p_agreement_id                  => l_agreement_id
    ,p_customer_id                   => l_customer_id
    ,p_customer_name                 => l_customer_name
    p_customer_num                   => l_customer_num
    ,p_agreement_num                 => l_agreement_num
    ,p_agreement_type                => l_agreement_type
    ,p_amount                        => l_amount
    ,p_term_id                       => l_term_id
,p_term_name                    => l_term_name
,p_revenue_limit_flag           => l_revenue_limit_flag
,p_expiration_date              => l_expiration_date
,p_description                  => l_description
,p_owned_by_person_id           => l_owned_by_person_id
,p_owned_by_person_name         => l_owned_by_person_name
,p_attribute_category           => l_attribute_category
,p_attribute1                   => l_attribute1
,p_attribute2                    => l_attribute2
,p_attribute3                   => l_attribute3
,p_attribute4                    => l_attribute4
,p_attribute5                    => l_attribute5
,p_attribute6                    => l_attribute6
,P_attribute7                    => l_attribute7
,p_attribute8                    => l_attribute8
,p_attribute9                    => l_attribute9
,p_attribute10                  => l_attribute10
,p_template_flag                => l_template_flag);

IF l_return_status != 'S'
THEN
  RAISE API_ERROR;
END IF;

-- LOAD_FUNDING  (loop for multiple Fundings )

FOR i IN 1..a LOOP
pa_agreement_pub.load_funding
    (p_api_version_number            => l_api_version_number
```

```
    ,p_init_msg_list                  => l_init_msg_list
    ,p_return_status                  => l_return_status
    ,p_pm_funding_reference           =>
l_funding_in(i).pm_funding_reference
 ,p_funding_id                        =>
l_funding_in(i).project_funding_id
    ,p_agreement_id                   =>
l_funding_in(i).agreement_id
    ,p_project_id                     =>
l_funding_in(i).project_id
    ,p_task_id                        =>
l_funding_in(i).task_id
    ,p_allocated_amount               =>
l_funding_in(i).allocated_amount
    ,p_date_allocated                 =>
l_funding_in(i).date_allocated
    ,p_attribute_category             =>
l_funding_in(i).attribute_category
    ,p_attribute1                     =>
l_funding_in(i).attribute1
    ,p_attribute2                     =>
l_funding_in(i).attribute2
    ,p_attribute3                     =>
l_funding_in(i).attribute3
    ,p_attribute4                     =>
l_funding_in(i).attribute4
    ,p_attribute5                     =>
l_funding_in(i).attribute5
    ,p_attribute6                     =>
l_funding_in(i).attribute6
    ,p_attribute7                     =>
l_funding_in(i).attribute7
    ,p_attribute8                     =>
l_funding_in(i).attribute8
    ,p_attribute9                     =>
l_funding_in(i).attribute9
    ,p_attribute10                    =>
l_funding_in(i).attribute10);

IF l_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;
END LOOP;
```

```
--EXECUTE_CREATE_AGREEMENT

pa_agreement_pub.execute_create_agreement

    ( p_api_version_number           =>
l_api_version_number,
    p_commit                         => l_commit,
    p_init_msg_list                  => l_init_msg_list,
    p_msg_count                      => l_msg_count,
    p_msg_data                       => l_msg_data
    p_return_status                  => l_return_status,
    p_pm_product_code                => l_pm_product_code,
    p_agreement_id_out               => l_agreement_id_out,
    p_customer_id_out                => l_customer_id_out);
IF l_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;

--FETCH_TASK

FOR l_funding_index in 1 ..a (loop for multiple Fundings)
LOOP
pa_agreement_pub.fetch_funding
      (p_api_version_number          => l_api_version_number
      ,p_init_msg_list               => l_init_msg_list
      ,p_return_status               => l_return_status
      ,p_funding_index               => l_funding_index
      ,p_funding_id                  => l_funding_id
      ,p_pm_funding_reference        =>
l_pm_funding_reference);
IF l_return_status != 'S'
THEN
  RAISE API_ERROR;
END IF;
END LOOP;
-------------------------------------------------
CLEAR_CREATE_AGREEMENT
pa_agreement_pub.clear_agreement;
-------------------------------------------------

IF l_return_status != 'S'
THEN
```

```
   RAISE API_ERROR;
END IF;
-- HANDLE EXCEPTIONS
EXCEPTION
WHEN API_ERROR THEN
    for i in 1..l_msg_count
    loop
       pa_interface_utils_pub.get_messages
                  (p_msg_data              => l_msg_data,
                   p_data                  => l_data,
                  p_msg_count        => l_msg_count,
                 p_msg_index_out           =>
l_msg_index_out);

     dbms_output.put_line ('error mesg '||l_data);

    end loop;

WHEN OTHERS THEN
  for i in 1..l_msg_count
   loop
         pa_interface_utils_pub.get_messages
                    (p_msg_data           => l_msg_data,
                     p_data               => l_data,
                    p_msg_count           => l_msg_count,
                   p_msg_index_out  => l_msg_index_out);

         dbms_output.put_line ('error mesg '||l_data),

   end loop;
END ;
```

Creating an Agreement Using a Composite Datatype API

The following sample PL/SQL code is a script that creates an agreement using the PA_AGREEMENT_PUB.CREATE_AGREEMENT, which uses composite datatypes.

If you create budgets using tools or products that do not support composite datatypes, see Creating an Agreement Using the Load–Execute–Fetch APIs: page 5 – 29.

```
DECLARE
  --variables needed for API standard parameters
l_api_version_number    NUMBER :=1.0;
l_commit                VARCHAR2(1):= 'F';
l_return_status         VARCHAR2(1);
l_init_msg_list         VARCHAR2(1);
l_msg_count             NUMBER;
l_msg_data              VARCHAR2(2000);
l_data                  VARCHAR2(2000);
l_msg_entity            VARCHAR2(100);
l_msg_entity_index      NUMBER;
l_msg_index             NUMBER;
l_msg_index_out         NUMBER;
l_encoded               VARCHAR2(1);
l_agreement_id_out      NUMBER;
l_customer_id-outNUMBER;
l_funding_id            NUMBER;

  --variables needed for Oracle Agreement specific
parameters
l_pm_product_codeVARCHAR2(25);
p_agreement_in_rec      pa_agreement_pub.Agreement_Rec._In_t
ype
p_agreement_out_rec     pa_agreement_pub.Agreement_Rec_Out_t
ype

  --variables needed for funding specific parameters
l_funding_type          pa_agreement_pub.funding_rec_in_type
;
l_agreement_in_rec      pa_agreement_pub.funding_in_tbl_type
;
l_funding_out_tblpa_agreement_pub.funding_out_tbl_type;

  --Funding Variables
   l_pm_funding_reference     VARCHAR2(25);
```

```
l_funding_rec          pa_agreement_pub.funding_rec_in_type
;
l_funding_in           pa_agreement_pub.funding_rec_in_type
;
l_funding_out          pa_agreement_pub.funding_rec_out_typ
e;

  -- Loop Variables;
a NUMBER
API_ERROR        EXCEPTION

--BEGIN

-- PRODUCT RELATED DATA

l_pm_product_code:='MSPROJECT';

--AGREEMENT DATA

p_agreement_in_rec.pm_agreement_reference   := 'AMGTEST1';
p_agreement_in_rec.agreement_id                    := Null;
p_agreement_in_rec.customer_id             := 21491;
p_agreement_in_rec.customer_num            := '1086';
p_agreement_in_rec.agreement_num           := 'AMGTEST1';
p_agreement_in_rec.agreement_type          := 'Contract';
p_agreement_in_rec.amount                  := 2000;
p_agreement_in_rec.term_id                 := 1000;
p_agreement_in_rec.term_name               := Null;
p_agreement_in_rec.revenue_limit_flag:= 'N';
p_agreement_in_rec.expiration_date         := Null;
p_agreement_in_rec.description             := Null;
p_agreement_in_rec.owned_by_person_id:= 1234;
p_agreement_in_rec.attribute_category:= Null;
p_agreement_in_rec.attribute1              := Null;
p_agreement_in_rec.attribute3              := Null;
p_agreement_in_rec.attribute4              := Null;
p_agreement_in_rec.attribute5              := Null;
p_agreement_in_rec.attribute6              := Null;
p_agreement_in_rec.attribute7              := Null;
p_agreement_in_rec.attribute8              := Null;
p_agreement_in_rec.attribute9              := Null;
p_agreement_in_rec.attribute10             := Null;
p_agreement_in_rec.template_flag           := 'N';
```

```
--FUNDING DATA
   a:= 1;
l_funding_rec.pm_funding_reference   := 'AMGTEST1FUN'
l_funding_rec.project_funding_id     := Null;
l_funding_rec.agreement_id           := Null;
l_funding_rec.project_id             := 7946;
l_funding_rec.task_id                := 10273;
l_funding_rec.allocated_amount       := 200;
l_funding_rec.date_allocated         := '27-DEC-01';
l_funding_rec.desc_flex_name         := Null;
l_funding_rec.attribute_category     := Null;
l_funding_rec.attribute1             := Null;
l_funding_rec.attribute2             := Null;
l_funding_rec.attribute3             := Null;
l_funding_rec.attribute4             := Null;
l_funding_rec.attribute5             := Null;
l_funding_rec.attribute6             := Null;
l_funding_rec.attribute7             := Null;
l_funding_rec.attribute8             := Null;
l_funding_rec.attribute9             := Null;
l_funding_rec.attribute10            := Null;

-- LOOP CONSTRUCT
    l_funding_in(a):= l_funding_rec;

-- CONSTRUCTING THE FUNDING TABLE
FOR i IN 1..a LOOP
l_funding_in(i).pm_funding_reference :=
l_funding_rec.pm_funding_reference
l_funding_in(i).project_funding_id   :=l_funding_rec.funding
_id;
l_funding_in(i).agreement_id         :=l_funding_rec.p_agree
ment_id;
l_funding_in(i).project_id           :=l_funding_rec._projec
t_id;
l_funding_in(i).task_id              :=
l_funding_rec.p_task_id;
l_funding_in(i).allocated_amount     :=l_funding_rec.p_alloc
ated_amount;
l_funding_in(i).date_allocated       :=l_funding_rec.p_date_
allocated;
l_funding_in(i).desc_flex_name       :=l_funding_rec.p_desc_
flex_name;
```

```
l_funding_in(i).attribute_category   :=l_funding_rec.p_attri
bute_category;
l_funding_in(i).attribute1           :=l_funding_rec.p_attri
bute1;
l_funding_in(i).attribute2           :=l_funding_rec.p_attri
bute2;
l_funding_in(i).attribute3           :=l_funding_rec.p_attri
bute3;
l_funding_in(i).attribute4           :=l_funding_rec.p_attri
bute4;
l_funding_in(i).attribute5           :=l_funding_rec.p_attri
bute5;
l_funding_in(i).attribute6           :=l_funding_rec.p_attri
bute6;
l_funding_in(i).attribute7           :=l_funding_rec.p_attri
bute7;
l_funding_in(i).attribute8           :=l_funding_rec.p_attri
bute8;
l_funding_in(i).attribute9           :=l_funding_rec.p_attri
bute9;
l_funding_in(i).attribute10          :=l_funding_rec.p_attri
bute10;
END LOOP;

-- 'CREATE_AGREEMENT

pa_agreement_pub.create_agreement
    ( p_api_version_number  => l_api_version_number
    ,p_commit           => l_commit
    ,p_init_msg_list    => l_init_msg_list
    ,p_msg_count        => l_msg_count
    ,p_msg_data         => l_msg_data
    ,p_return_status    => l_return_status
    ,p_pm_product_code  => l_pm_product_code
    ,p_agreement_in_rec => p_agreement_in_rec
    ,p_agreement_out_rec=> p_agreement_out_rec
    ,p_funding_in_tbl   => l_funding_in
    ,p_funding_out_tbl  => l_funding_out);
IF l_return_status != 'S'
THEN
RAISE API_ERROR;
END IF;

--HANDLE EXCEPTIONS
```

```
EXCEPTION
WHEN API_ERROR THEN
for i in 1..1_msg_count
loop
   pa_interface_utils_pub.get_messages
          (p_msg_date                => l_msg_date
          ,p_data                    => l_data
          ,p_msg_count               => l_msg_count
          ,p_msg_index_out           => l_msg_index_out)

   dbms_output.put_line ('error mesg' 1_data)

   end loop;
if i = 1 THEN

WHEN OTHERS THEN
pa_interface_utils_pub.get_messages
          (p_msg_data                => l_msg_data
          ,p_data                    => l_data
          ,p_msg_count               => l_msg_count
          ,p_msg_index_out           => l_msg_index_out);

   dbms_output.put_line ('error mesg'1_data)
END;
/
```

Event APIs

The event APIs provide an open interface for external systems to insert, update, and delete events.

Event API Procedures

The procedures discussed in this section are listed below. The procedures are located in the public API package PA_EVENT_PUB.

Event API Procedure Definitions

This section contains detailed description of the event APIs.

CREATE_EVENT

This API creates an event or a set of events.

The following table shows the parameters for CREATE_EVENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_EVENT_IN_TBL	IN	EVENT_IN_TBL_TYPE	No	The reference code that uniquely identifies the event input record in Oracle projects
P_EVENT_OUT_TBL	OUT	EVENT_OUT_TBL_TYPE		The reference code that uniquely identifies the event output record in Oracle Projects
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

DELETE_EVENT

This API deletes an event.

The following table shows the parameters for DELETE_EVENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)

Name	Usage	Type	Req?	Description
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_EVENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the event in the external system
P_EVENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the event in Oracle Projects
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

UPDATE_EVENT

This API updates an event or set of events.

The following table shows the parameters for UPDATE_EVENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	YES	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG__LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_EVENT_IN_TBL	IN	EVENT_IN_TBL_TYPE	No	The reference code that uniquely identifies the event input record in Oracle projects
P_EVENT_OUT_TBL	OUT	EVENT_OUT_TBL_TYPE		The reference code that uniquely identifies the event output record in Oracle Projects
P_MSG_COUNT	OUT	NUMBER	Yes	API standard
P_MSG_DATA	OUT	VARCHAR2(2000)	Yes	API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)	Yes	API standard

INIT_EVENT

This API sets the global tables used by the Load–Execute–Fetch procedures that create a new event or update an existing event. This API has no parameters.

LOAD_EVENT

This API loads an event to a PL/SQL record.

The following table shows the parameters for LOAD_EVENT.

Name	Usage	Type	Req?	Description
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_PM_EVENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the event in the external system
P_TASK_NUMBER	IN	VARCHAR2(25)	Yes	The number that identifies the task in Oracle Projects
P_EVENT_NUMBER	IN	NUMBER	Yes	The number that identifies the event
P_EVENT_TYPE	IN	VARCHAR2(30)	Yes	The event type that classifies the event
P_DESCRIPTION	IN	VARCHAR2(250)	No	Description of the event
P_BILL_HOLD_FLAG	IN	VARCHAR2(1)	No	Indicator that the event is held from invoicing
P_COMPLETION_DATE	IN	DATE	No	The date on which the event is complete and on or after which the event is processed for revenue accrual and/or invoicing
P_DESC_FLEX_NAME	IN	VARCHAR2(240)	No	Descriptive flexfield name
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Descriptive flexfield category
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2(150)	No	Descriptive flexfield attribute
P_PROJECT_NUMBER	IN	VARCHAR2(25)	No	The project number associated with the event
P_ORGANIZATION_NAME	IN	VARCHAR2(240)	No	The organization associated with the event
P_INVENTORY_ORG_NAME	IN	VARCHAR2(240)	No	The inventory organization associated with the event

Name	Usage	Type	Req?	Description
P_INVENTORY_ITEM_ID	IN	NUMBER	No	The inventory item ID associated with the event
P_QUANTITY_BILLED	IN	NUMBER	No	The quantity billed
P_UOM_CODE	IN	VARCHAR2(3)	No	The unit of measure
P_UNIT_PRICE	IN	NUMBER	No	The unit price
P_REFERENCE1 through P_REFERENCE10	IN	VARCHAR2(240)	No	Reference column
P_BILL_TRANS_CURRENCY_CODE	IN	VARCHAR2(15)	No	Billing transaction currency code
P_BILL_TRANS_BILL_AMOUNT	IN	NUMBER	No	Billing transaction billing amount
P_BILL_TRANS_REV_AMOUNT	IN	NUMBER	No	Billing transaction revenue amount
P_PROJECT_RATE_TYPE	IN	VARCHAR2(30)	No	Event currency to project currency exchange rate type
P_PROJECT_RATE_DATE	IN	DATE	No	Event currency to project currency exchange rate date
P_PROJECT_EXCHANGE_RATE	IN	NUMBER	No	Event currency to project currency exchange rate
P_PROJFUNC_RATE_TYPE	IN	VARCHAR2(30)	No	Event currency to project functional currency exchange rate type
P_PROJFUNC_RATE_DATE	IN	DATE	No	Event currency to project functional currency exchange rate date
P_PROJFUNC_EXCHANGE_RATE	IN	NUMBER	No	Event currency to project functional currency exchange rate
P_FUNDING_RATE_TYPE	IN	VARCHAR2(30)	No	Funding currency to project currency exchange rate type
P_FUNDING_RATE_DATE	IN	DATE	No	Funding currency to project currency exchange rate date
P_FUNDING_EXCHANGE_RATE	IN	NUMBER	No	Funding currency to project currency exchange rate
P_ADJUSTING_REVENUE_FLAG	IN	VARCHAR2(1)	No	Indicates revenue adjustment
P_EVENT_ID	IN	NUMBER	No	Identifier of the event
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

EXECUTE_CREATE_EVENT

This API creates an event using the data which is stored in the global tables during the Load phase.

The following table shows the parameters for EXECUTE_CREATE_EVENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = F)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_EVENT_ID_OUT	OUT	NUMBER	Yes	The reference code that uniquely identifies the event in Oracle Projects
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

EXECUTE_UPDATE_EVENT

This API updates event data using the information stored in the global tables during the Load phase.

The following table shows the parameters for EXECUTE_UPDATE_EVENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (Default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (Default = F)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

FETCH_EVENT

This API gets the return_status that was returned during creation of an event and stored in a global PL/SQL table.

The following table shows the parameters for FETCH_EVENT.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	YES	API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_EVENT_REFERENCE	IN	VARCHAR2		The reference code that uniquely identifies the event in the external system
P_EVENT_ID_OUT	OUT	NUMBER		The reference code that uniquely identifies the event in Oracle Projects.
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

CLEAR_EVENT

This API clears the globals that were set up during initialization.

Parameters: None

CHECK_DELETE_EVENT_OK

This API checks whether an event can be deleted.

The following table shows the parameters for CHECK_DELETE_EVENT_OK.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = F)
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = F)
P_PM_PRODUCT_CODE	IN	VARCHAR2(30)	Yes	The identifier of the external project management system from which the project was imported
P_PM_EVENT_REFERENCE	IN	VARCHAR2(25)	Yes	The reference code that uniquely identifies the event in the external system
P_EVENT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the event in Oracle Projects.
P_DEL_EVENT_OK_FLAG	OUT	VARCHAR2(1)		Boolean flag for deleting event
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		APi standard

CHAPTER

6

Oracle Project Management APIs

This chapter describes how to implement APIs for:

- Budget information
- Project status information

Budget APIs

Budgets track the time and resources that you expect to use to complete a project or task. Use your external system to prepare your budget, and then use Budget APIs to interface the budget and budget line into Oracle Projects. Oracle Projects then generates a budget based on the resource budgets and rates stored in the external system. You can interface multiple budget versions to Oracle Projects and baseline them as needed.

> **Note:** When you call a budget API that requires a project identifier, you must pass either the P_PA_PROJECT_ID or the P_PM_PROJECT_REFERENCE parameter to identify the project. When you call a budget API that requires a resource list identifier, you must pass either the P_RESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID parameter to identify the resource list.

Budget API Views

The following table lists the views that provide parameter data for the budget APIs. For detailed description of the views, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

View	Description
PA_BASE_BUDGET_BY_GL_PERIOD_V	Most recent baselined budget amounts by GL period
PA_BASE_BUDGET_BY_PA_PERIOD_V	Most recent baselined budget amounts by PA period
PA_BUDGET_CHANGE_REASON_V	Retrieves budget change reason codes
PA_BUDGET_ENTRY_METHODS_V	Retrieves budget entry methods
PA_BUDGET_STATUS_CODES_V	Retrieves budget status codes
PA_BUDGET_TYPES_V	Retrieves budget types

Table 6 – 1 Budget API views (Page 1 of 2)

View	Description
PA_ORIG_BUDGET_BY_GL_PERIOD_V	Original budget amounts by GL perio.
PA_ORIG_BUDGET_BY_PA_PERIOD_V	Original budget amounts by PA period

Table 6 – 1 Budget API views (Page 2 of 2)

Budget API Procedures

The procedures discussed in this section are listed below. The procedures are located in the public API package PA_PROJECT_PUB.

- Budget and Budget Line Procedures
  - ADD_BUDGET_LINE: page 6 – 4
  - BASELINE_BUDGET: page 6 – 7
  - CALCULATE_AMOUNTS: page 6 – 8
  - CREATE_DRAFT_BUDGET: page 6 – 10
  - DELETE_BUDGET_LINE: page 6 – 17
  - DELETE_DRAFT_BUDGET: page 6 – 19
  - UPDATE_BUDGET: page 6 – 20
  - UPDATE_BUDGET_LINE: page 6 – 23
- Load–Execute–Fetch Procedures
  - CLEAR_BUDGET: page 6 – 26
  - EXECUTE_CALCULATE_AMOUNTS: page 6 – 27
  - EXECUTE_CREATE_DRAFT_BUDGET: page 6 – 29
  - EXECUTE_UPDATE_BUDGET: page 6 – 32
  - FETCH_BUDGET_LINE: page 6 – 33
  - FETCH_CALCULATE_AMOUNTS: page 6 – 34
  - INIT_BUDGET: page 6 – 35
  - INIT_CALCULATE_AMOUNTS: page 6 – 35
  - LOAD_BUDGET_LINE: page 6 – 35

Budget API Procedure Definitions

This section contains description of the budget APIs, including business rules and parameters.

ADD_BUDGET_LINE

ADD_BUDGET_LINE is a PL/SQL procedure used to add a budget line to a working budget in Oracle Projects for a given project and budget type.

Business Rules

Note: This API does not support the Web–based budget user interface.

- After you use ADD_BUDGET_LINE to create a draft budget and budget lines, save the data to the database before calling the API BASELINE_BUDGET. (A draft budget requires approval before you can baseline it.) For a revenue budget, enter the funding in Oracle Projects before you baseline the budget.
- We establish the following links between information stored in your external system and certain information in Oracle Projects, so you can pass the following parameters instead of their corresponding Oracle Projects identification codes.
  - For budgets:

    P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
    P_RESOURCE_LIST_NAME links to P_RESOURCE_LIST_ID.
  - For budget lines:

    P_PM_TASK_REFERENCE links to P_PA_TASK_ID.
    P_RESOURCE_ALIAS links to P_RESOURCE_LIST_MEMBER_ID.
- The following pairs of parameters may both have NULL values if the budget is not categorized by resources, as defined by the budget entry method:
  - P_RESOURCE_LIST_NAME and P_RESOURCE_LIST_ID
  - RESOURCE_ALIAS and RESOURCE_LIST_MEMBER_ID

- Specify values for the parameters PA_TASK_ID or PM_TASK_REFERENCE only when budgeting by tasks, as defined by the budget entry method.
- Specify values for the parameter PERIOD_NAME only when budgeting by PA or GL period, as defined by the budget entry method.
- If you budget by PA or GL period and do not provide a period name, Oracle Projects uses the budget start and end dates to select a valid period name from the database. If Oracle Projects fails to retrieve a valid period name, the API will abort.
- The task level at which you pass budget information should correspond to the level specified in the budget entry method. For example, if the budget entry method specifies that you can enter a budget only at the lowest task level, then ADD_BUDGET_LINE passes only lowest tasks.
- When the budget entry method (BEM) flags shown in the following table are set to N, do not pass the related parameters.

BEM Flag	***Related Parameter***
COST_QUANTITY_FLAG	QUANTITY
RAW_COST_FLAG	RAW_COST
BURDENED_COST_FLAG	BURDENED_COST
REV_QUANTITY_FLAG	QUANTITY
REVENUE_FLAG	REVENUE

Table 6 – 2 Parameters not to Pass if BEM Flags Are Set to N (Page 1 of 1)

- You can add a budget line only to a budget with a status of Working.

The following table shows the parameters for ADD_BUDGET_LINE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard

Name	Usage	Type	Req?	Description
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_PA_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	Alias of a resource
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	The identification code of the resource
P_BUDGET_START_DATE	IN	DATE	No	Start date of a budget line
P_BUDGET_END_DATE	IN	DATE	No	End date of a budget line
P_PERIOD_NAME	IN	VARCHAR2(30)	No	GL or PA period name
P_DESCRIPTION	IN	VARCHAR2(255)		(currently unavailable)
P_RAW_COST	IN	NUMBER	No	Budgeted raw cost amount
P_BURDENED_COST	IN	NUMBER	No	Budgeted burdened cost amount
P_REVENUE	IN	NUMBER	No	Budgeted revenue amount
P_QUANTITY	IN	NUMBER	No	Budgeted quantity
P_PM_BUDGET_LINE_REFERENCE	IN	VARCHAR2(30)	No	The reference code that identifies the budget line on the client side
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget line descriptive flexfield

BASELINE_BUDGET

BASELINE_BUDGET is a PL/SQL procedure used to baseline an existing budget in Oracle Projects for a given project and budget type.

Business Rules

- The following parameters are used only for the Web–based user interface. For budgets that do not use the Web–based interface, these parameters are not applicable:
  - P_FIN_PLAN_TYPE_ID
  - P_FIN_PLAN_TYPE_NAME
  - P_VERSION_TYPE
- You must set up funding in Oracle Projects before you can baseline a revenue budget.
- If you have not yet submitted a budget, Oracle Projects submits it automatically before baselining it.
- You can submit a budget only if it contains budget lines.
- If no value (or an invalid value) is passed for the parameter P_MARK_AS_ORIGINAL, the default is N. When you baseline a budget for the first time, the P_MARK_AS_ORIGINAL is set to Y.

The following table shows the parameters for BASELINE_BUDGET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_WORKFLOW_STARTED	OUT	VARCHAR2(1)		Shows if a workflow has been started (Y or N)
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system

Name	Usage	Type	Req?	Description
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_MARK_AS_ORIGINAL	IN	VARCHAR2(1)	No	Mark as original
P_FIN_PLAN_TYPE_ID	IN	NUMBER	No	The unique identifier of the plan type being used for the creation of a plan version
P_FIN_PLAN_TYPE_NAME	IN	VARCHAR2(150)	No	The plan type name. You must supply a valid value for either p_fin_plan_type_id or p_fin_plan_type_name.
P_VERSION_TYPE	IN	VARCHAR2(30)	No	The version type. A value for this parameter is required when the plan type is set up to plan cost and revenue versions separately. In other cases, if the value is not passed, then it is derived automatically.

CALCULATE_AMOUNTS

Using the PA_CLIENT_EXTN_BUDGET extension, you can use the public API CALCULATE_AMOUNTS to recalculate raw cost, burdened cost, and revenue amounts for existing budget lines. If P_UPDATE_DB_FLAG is set to Y, then the budget lines for the specified project will be updated upon the successful execution of this API.

Business Rules

- The following parameters are used only for the Web–based user interface. For budgets that do not use the Web–based interface, these parameters are not applicable:
  - P_BUDGET_VERSION_ID
  - P_FIN_PLAN_TYPE_ID
  - P_FIN_PLAN_TYPE_NAME
  - P_VERSION_TYPE
  - P_BUDGET_VERSION_NUMBER
- Since this API calls the PA_CLIENT_EXTN_BUDGET extension, you must modify the extension to calculate the amounts you want.

- To recalculate the corresponding amount, pass an uppercase Y for each calculation flag.
- Regardless of its update status, CALCULATE_AMOUNTS returns one row of amounts for each budget line it reads.
- To update the budget lines for a project with the amounts generated from CALCULATE_AMOUNTS, set P_UPDATE_DB_FLAG to an uppercase Y.

The following table shows the parameters for CALCULATE_AMOUNTS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(25)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_CALC_RAW_COST_YN	IN	VARCHAR2(1)	No	Calculate raw cost (Y = Yes, N = No)
P_CALC_BURDENED_COST_YN	IN	VARCHAR2(1)	No	Calculate burdened cost (Y = Yes, N = No)
P_CALC_REVENUE_YN	IN	VARCHAR2(1)	No	Calculate revenue (Y = Yes, N = No)
P_UPDATE_DB_FLAG	IN	VARCHAR2(1)	No	Update budget lines (Y = Yes, N = No)
P_CALC_BUDGET_LINES_OUT	IN	TABLE OF RECORD		
P_BUDGET_VERSION_ID	IN	NUMBER(15)	No	The system-generated number that uniquely identifies the budget version.
P_FIN_PLAN_TYPE_ID	IN	NUMBER	No	The unique identifier of the plan type being used for the creation of a plan version
P_FIN_PLAN_TYPE_NAME	IN	VARCHAR2(150)	No	The plan type name. You must supply a valid value for either p_fin_plan_type_id or p_fin_plan_type_name.

Name	Usage	Type	Req?	Description
P_VERSION_TYPE	IN	VARCHAR2(30)	No	The version type. A value for this parameter is required when the plan type is set up to plan cost and revenue versions separately. In other cases, if the value is not passed, then it is derived automatically.
P_BUDGET_VERSION_NUMBER	IN	NUMBER(15)	No	The version number for the budget
PA_TASK_ID	OUT	NUMBER		The reference code that uniquely identifies the task within a project in Oracle Projects
PM_TASK_REFERENCE	OUT	VARCHAR2(30)		The reference code that uniquely identifies the task in the external system
RESOURCE_ALIAS	OUT	VARCHAR2(30)		Alias of a resource
RESOURCE_LIST_MEMBER_ID	OUT	NUMBER		The identification code of the resource
BUDGET_START_DATE	OUT	DATE		Start date of a budget
BUDGET_END_DATE	OUT	DATE		End date of a budget
PERIOD_NAME	OUT	VARCHAR2(30)		PA or GL period name
CALCULATED_RAW_COST	OUT	NUMBER		Calculated raw cost
CALCULATED_BURDENED_COST	OUT	NUMBER		Calculated burdened cost
CALCULATED_REVENUE	OUT	NUMBER		Calculated revenue
QUANTITY	OUT	NUMBER		Quantity
RETURN_STATUS	OUT	VARCHAR2(1)		API standard

CREATE_DRAFT_BUDGET

CREATE_DRAFT_BUDGET is a PL/SQL procedure used to create a draft budget and its budget lines in Oracle Projects for a given project, using a selected budget type and budget entry method.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes: page 2 – 19.

Business Rules

- Some parameters are used only for the Web–based user interface. For budgets that do not use the Web–based interface, thefollowing types of parameters are not applicable:
  - Plan type and plan code parameters
  - Version type parameters

  - P_TIME_PHASED_CODE
  - P_PLAN_IN_MULTI_CURR_FLAG
  - Currency attributes
  - Flags for raw cost, burdened cost, revenue, quantity planning, create current working version, replace current working version
  - P_USING_RESOURCE_LISTS_FLAG

- A draft budget requires approval before you can baseline it. After you use this API to create a draft budget and budget lines, save the data to the database before calling the API BASELINE_BUDGET. For a revenue budget, enter the funding in Oracle Projects before you can baseline the budget.
- We establish the following links between information stored in your system and certain information in Oracle Projects, so you can pass the following parameters instead of their corresponding Oracle Projects identification codes.
  - For budgets:
    P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
    P_RESOURCE_LIST_NAME links to P_RESOURCE_LIST_ID.
  - For budget lines
    P_PM_TASK_REFERENCE links to P_PA_TASK_ID.
    P_RESOURCE_ALIAS links to P_RESOURCE_LIST_MEMBER_ID.
- Products that call budget APIs must specify their respective product codes. Oracle Projects predefines product codes and provides these codes to the appropriate vendors.
- The following pairs of parameters can both have NULL values if the budget is not categorized by resources, as defined by the budget entry method:
  - P_RESOURCE_LIST_NAME and P_RESOURCE_LIST_ID
  - RESOURCE_ALIAS and RESOURCE_LIST_MEMBER_ID
- You can specify a value for the PA_TASK_ID or PM_TASK_REFERENCE parameter only when budgeting by tasks, as defined by the budget entry method.
- You can specify a value for the PERIOD_NAME parameter only when budgeting by PA or GL period, as defined by the budget entry method.

- If you budget by PA or GL period and do not provide a period name, Oracle Projects uses the budget start and end dates to select a valid period name from the database. If Oracle Projects fails to retrieve a valid period name, the API will abort.
- When budgeting by date range, you must provide the budget start and end dates. These dates may not overlap for a certain resource assignment.
- The task level at which you pass budget information should correspond to the level specified in the budget entry method. For example, if the budget entry method specifies that you can enter a budget only at the lowest task level, then this API passes only lowest tasks.
- When the budget entry method (BEM) flags shown in the following table are set to N, do not pass the related parameters.

BEM Flag	***Related Parameter***
COST_QUANTITY_FLAG	QUANTITY
RAW_COST_FLAG	RAW_COST
BURDENED_COST_FLAG	BURDENED_COST
REV_QUANTITY_FLAG	QUANTITY
REVENUE_FLAG	REVENUE

Table 6 – 3 Parameters not to Pass if BEM Flags Are Set to N (Page 1 of 1)

- Your budget entry method must reflect the needs of your external system.
- You can specify values for the parameters P_RAW_COST and P_BURDENED_COST amounts only for a cost budget, as defined by the budget type.
- You can specify a value for the parameter P_REVENUE_AMOUNT only for a revenue budget, as defined by the budget type.
- Passing the PL / SQL table P_BUDGET_LINES_TBL is optional. A draft budget does not require you to create budget lines simultaneously.
- If a draft budget already exists for a project and budget type, creating a new draft budget deletes the existing budget and budget lines.

The following table shows the parameters for CREATE_DRAFT_BUDGET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PM_BUDGET_REFERENCE	IN	VARCHAR2(30)	No	The reference code of the budget on the client side
P_BUDGET_VERSION_NAME	IN	VARCHAR2	No	The user-defined name for the budget version
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_CHANGE_REASON_CODE	IN	VARCHAR2(30)	No	The reference code that identifies the change reason
P_DESCRIPTION	IN	VARCHAR2(255)	No	Description of the budget
P_ENTRY_METHOD_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget entry method
P_RESOURCE_LIST_NAME	IN	VARCHAR2(60)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	The identification code of the resource list
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget descriptive flexfield
P_BUDGET_LINES_IN	IN	TABLE OF RECORD		
TXN_CURRENCY_CODE	IN	VARCHAR2(15)	Yes	The transaction currency code for the budget line
CHANGE_REASON_CODE	IN	VARCHAR2(30)	No	The reference code that identifies the change reason
PROJFUNC_COST_EXCHANGE_RATE	IN	NUMBER	No	Therate for converting cost amounts from the transaction currency to the project functional currency

Name	Usage	Type	Req?	Description
PROJFUNC_REV_EXCHANGE_RATE	IN	NUMBER	No	The rate for converting revenue amounts from the transaction currency to the project functional currency.
PROJCOST_COST_EXCHANGE_RATE	IN	NUMBER	No	The rate for converting cost amounts from the transaction currency to the project currency.
PROJCOST_REV_EXCHANGE_RATE	IN	NUMBER	No	The rate for converting revenue amounts from the transaction currency to the project currency
PA_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects
PM_TASK_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
RESOURCE_ALIAS	IN	VARCHAR2(30)	No	The alias of a resource
RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	The identification code of the resource
BUDGET_START_DATE	IN	DATE	No	Start date of budget line
BUDGET_END_DATE	IN	DATE	No	End date of budget line
PERIOD_NAME	IN	VARCHAR2(30)	No	GL or PA period name
DESCRIPTION	IN	VARCHAR2(255)	No	(currently unavailable)
RAW_COST	IN	NUMBER	No	Budgeted raw cost amount
BURDENED_COST	IN	NUMBER	No	Budgeted burdened cost amount
REVENUE	IN	NUMBER	No	Budgeted revenue amount
QUANTITY	IN	NUMBER	No	Budgeted quantity
PM_PRODUCT_CODE	IN	VARCHAR2(30)	No	The product code of the vendor of the external system
PM_BUDGET_LINE_REFERENCE	IN	VARCHAR2(30)	No	The reference code that identifies the budget line on client side
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
ATTRIBUTE1 through ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget line descriptive flexfield
P_FIN_PLAN_TYPE_ID	IN	NUMBER	No	The unique identifier of the plan type being used for the creation of a plan version
P_FIN_PLAN_TYPE_NAME	IN	VARCHAR2(150)	No	The plan type name. You must supply a valid value for either p_fin_plan_type_id or p_fin_plan_type_name

Name	Usage	Type	Req?	Description
P_VERSION_TYPE	IN	VARCHAR2(30)	No	The version type. A value for this parameter is required when the plan type is set up to plan cost and revenue versions separately. In other cases, if the value is not passed, then it is derived automatically.
P_FIN_PLAN_LEVEL_CODE	IN	VARCHAR2(30)	No	The planning level for the plan version. Valid values are P (project-level planning), T (top task-level planning), M (mixed-level planning - top and lowest tasks), and L (lowest task-level planning).
P_TIME_PHASED_CODE	IN	VARCHAR2(30)	No	The time phasing option. Valid values are P (planning by PA periods), G (planning by GL periods), R (planning by date range), and N (None-implies that dates are derived from the project or task dates.
P_PLAN_IN_MULTI_CURR_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether the plan version uses multiple transaction currencies
P_PROJFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting cost amounts from the transaction currency to the project functional currency
P_PROJFUNC_COST_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting cost amounts from transaction currency to project functional currency
P_PROJFUNC_COST_RATE_DATE	IN	DATE	No	The rate date for converting cost amounts from transaction currency to project functional currency
P_PROJFUNC_REV_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting revenue amounts from the transaction currency to the project functional currency
P_PROJFUNC_REV_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting revenue amounts from transaction currency to project functional currency
P_PROJFUNC_REV_RATE_DATE	IN	DATE	No	The rate date for converting revenue amounts from transaction currency to project functional currency
P_PROJECT_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting cost amounts from the transaction currency to the project currency

Name	Usage	Type	Req?	Description
P_PROJECT_COST_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting cost amounts from transaction currency to project currency
P_PROJECT_COST_RATE_DATE	IN	DATE	No	The rate date for converting cost amounts from transaction currency to project currency
P_PROJECT_REV_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting revenue amounts from the transaction currency to the project currency
P_PROJECT_REV_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting revenue amounts from transaction currency to project currency
P_PROJECT_REV_RATE_DATE	IN	DATE	No	The rate date for converting revenue amounts from transaction currency to project currency
P_RAW_COST_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether raw cost can be planned for the plan version
P_BURDENED_COST_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether burdened cost can be planned for the plan version
P_REVENUE_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether revenue can be planned for the plan version
P_COST_QTY_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether quantity can be planned for the plan version of version type cost
P_REVENUE_QTY_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether quantity can be planned for the plan version of version type revenue
P_ALL_QTY_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether quantity can be planned when cost and revenue are planned together in a single plan version
P_CREATE_NEW_CURR_WORKING_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether a current working version should be created. This parameter is required only if the budget uses the Web-based user interface.

Name	Usage	Type	Req?	Description
P_REPLACE_CURRENT_WORKING_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether the current working version should be deleted and the newly created version marked as the Current Working version. This parameter is required only if the budget uses the Web-based user interface.
P_USING_RESOURCE_LISTS_FLAG	IN	VARCHAR2(1)	No	When a plan amount is not classified using a resource list, then N (no) must be entered as the parameter value. This parameter is required only if the budget uses the Web-based user interface.
P_BUDGET_LINES_OUT	OUT	TABLE OF RECORD		
RETURN_STATUS	OUT	VARCHAR2(1)		Return status

DELETE_BUDGET_LINE

DELETE_BUDGET_LINE is a PL/SQL procedure used to delete a budget line from a working budget in Oracle Projects for a given project and budget type.

Business Rules

Note: This API does not support the Web–based budget user interface.

This rule applies to the budget status that supports budget line deletion:

- You can delete only budget lines from working budgets. You cannot delete budget lines from baselined budgets.

This rule applies to the budget start date and period name:

- If values for P_START_DATE and P_PERIOD_NAME are not passed or are both passed as NULL, deleting a budget line deletes **all** the budget lines for the task/resource combination.

These rules apply to the budget entry method:

- Depending on the budget entry method, this API may require that you pass task and/or resource data.
- If budget APIs have passed no task data, Oracle Projects assumes that the budget entry method has specified

uncategorized budgeting (budgets not tracked by resource) and project–level budgeting.

- If APIs pass both the P_START_DATE and the P_PERIOD_NAME to Oracle Projects, Oracle Projects uses the latter.

The following table shows the parameters for DELETE_BUDGET_LINE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_PA_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	Alias of a resource
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	The identification code of the resource
P_START_DATE	IN	DATE	No	The identification code of the budget line
P_PERIOD_NAME	IN	VARCHAR2(30)	No	The identification code of the budget line; overrules P_START_DATE

DELETE_DRAFT_BUDGET

DELETE_DRAFT_BUDGET is a PL/SQL procedure used to delete a working budget in Oracle Projects for a given project and budget type.

Business Rules

- You can delete working budgets only. You cannot delete baselined or submitted budgets.
- When you delete a budget, you also delete its budget lines and resource assignments.

The following table shows the parameters for DELETE_DRAFT_BUDGET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_FIN_PLAN_TYPE_NAME	IN	VARCHAR2(150)	No	The plan type name. You must supply a valid value for either p_fin_plan_type_id or p_fin_plan_type_name.
P_FIN_PLAN_TYPE_ID	IN	NUMBER	No	The unique identifier of the plan type used to create the plan version

Name	Usage	Type	Req?	Description
P_VERSION_NUMBER	IN	NUMBER(15)	No	The version number for the budget
P_VERSION_TYPE	IN	VARCHAR2(30)	No	The version type. A value for this parameter is required when the plan type is set up to plan cost and revenue versions separately. In other cases, if the value is not passed, then it is derived automatically.

UPDATE_BUDGET

UPDATE_BUDGET is a PL/SQL procedure used to update the working budget with its budget lines in Oracle Projects for a given project. This API updates existing budget lines or inserts new budget lines, depending on whether the budget lines already exist.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes: page 2 – 19.

Business Rules

Note: This API does not support the Web–based budget user interface.

- A draft budget requires approval before you can baseline it. After you use this API to create a draft budget and budget lines, save the data to the database before calling the API BASELINE_BUDGET. For a revenue budget, you must enter the funding in Oracle Projects before you can baseline the budget.
- We establish links between information stored in your external system and certain information in Oracle Projects, so you can pass the following parameters instead of their corresponding Oracle Projects identifiers.
  - For budgets:
    P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
    P_RESOURCE_LIST_NAME links to P_RESOURCE_LIST_ID.
  - For budget lines:
    P_PM_TASK_REFERENCE links to P_PA_TASK_ID.
    P_RESOURCE_ALIAS links to P_RESOURCE_LIST_MEMBER_ID.

- The following pairs of parameters can both have NULL values if the budget is not categorized by resources, as defined by the budget entry method:
  - P_RESOURCE_LIST_NAME and P_RESOURCE_LIST_ID
  - RESOURCE_ALIAS and RESOURCE_LIST_MEMBER_ID
- You can specify values for the parameters PA_TASK_ID or PM_TASK_REFERENCE only when budgeting by tasks, as defined by the budget entry method.
- You can specify values for the parameter PERIOD_NAME only when budgeting by PA or GL period, as defined by the budget entry method.
- If you budget by PA or GL period and do not provide a period name, Oracle Projects uses the budget start date and budget end date to select a valid period name from the database. If Oracle Projects fails to retrieve a valid period name, the API will abort.
- The task level at which you pass budget information should correspond to the level specified in the budget entry method. For example, if the budget entry method specifies that you can enter a budget only at the lowest task level, then this API passes only lowest tasks.
- When the budget entry method flags shown in the following table are set to N, do not pass the related parameters.

BEM Flag	***Related Parameter***
COST_QUANTITY_FLAG	QUANTITY
RAW_COST_FLAG	RAW_COST
BURDENED_COST_FLAG	BURDENED_COST
REV_QUANTITY_FLAG	QUANTITY
REVENUE_FLAG	REVENUE

Table 6 – 4 Parameters not to Pass if BEM Flags Are Set to N (Page 1 of 1)

- You can add a budget line only to a budget with a status of Working.
- You cannot update a submitted budget.
- You can use this API only to update or add budget lines. To delete existing budget lines, use DELETE_BUDGET_LINE.

- Oracle Projects identifies a budget line by its budget start date, so you cannot update the budget start date.
- You can update only the budget header P_CHANGE_REASON_CODE and P_DESCRIPTION parameters.
- You can update only the following budget line parameters:
  - DESCRIPTION
  - RAW_COST
  - BURDENED_COST
  - REVENUE
  - QUANTITY
- You can pass flexfield parameters for both budget headers and budget lines. However, you can currently use flexfield parameters only to create new budget line rows.
- When passed, the parameters marked with an asterisk (*) in the following table identify the budget and budget line.

The following table shows the parameters for UPDATE_BUDGET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID*	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE*	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE*	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_CHANGE_REASON_CODE	IN	VARCHAR2(30)	No	The reference code that identifies the change reason
P_DESCRIPTION	IN	VARCHAR2(255)	No	Description of the budget
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields

Name	Usage	Type	Req?	Description
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget descriptive flexfield
P_BUDGET_LINES_IN	IN	TABLE OF RECORD		
PA_TASK_ID*	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects
PM_TASK_REFERENCE*	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
RESOURCE_ALIAS*	IN	VARCHAR2(30)	No	Alias of a resource uniquely identifies the task in the external system
RESOURCE_LIST_MEMBER_ID*	IN	NUMBER	No	The identification code of the resource
BUDGET_START_DATE*	IN	DATE	No	Start date of budget line
BUDGET_END_DATE*	IN	DATE	No	End date of budget line
PERIOD_NAME*	IN	VARCHAR2(30)	No	GL or PA period name
DESCRIPTION	IN	VARCHAR2(255)	No	(currently unavailable)
RAW_COST	IN	NUMBER	No	Budgeted raw cost amount
BURDENED_COST	IN	NUMBER	No	Budgeted burdened cost amount
REVENUE	IN	NUMBER	No	Budgeted revenue amount
QUANTITY	IN	NUMBER	No	Budgeted quantity
PA_PRODUCT_CODE	IN	VARCHAR2(30)	No	The product code of the vendor of the external system
PM_BUDGET_LINE_REFERENCE	IN	VARCHAR2(30)	No	Reference code that identifies the budget line on the client side
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
ATTRIBUTE1 through ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget line descriptive flexfield
P_BUDGET_LINES_OUT	OUT	TABLE OF RECORD		Return status
RETURN_STATUS	OUT	VARCHAR2(1)	No	

UPDATE_BUDGET_LINE

UPDATE_BUDGET_LINE is a PL/SQL procedure used to update an existing budget line of a working budget in Oracle Projects for a given project and budget type.

Business Rules

Note: This API does not support the Web–based budget user interface.

- A draft budget requires approval before you can baseline it. After you use this API to create a draft budget and budget lines, you must save the data to the database before calling the API BASELINE_BUDGET. For a revenue budget, enter the funding in Oracle Projects before you baseline the budget.
- We establish links between information stored in your external system and certain information in Oracle Projects, so you can pass the following parameters instead of their corresponding Oracle Projects identification codes.
  - For budgets:
    P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
    P_RESOURCE_LIST_NAME links to
    P_RESOURCE_LIST_ID.
  - For budget lines:
    P_PM_TASK_REFERENCE links to P_PA_TASK_ID.
    P_RESOURCE_ALIAS links to
    P_RESOURCE_LIST_MEMBER_ID.
- The following pairs of parameters can both have NULL values if the budget is not categorized by resources, as defined by the budget entry method:
  - P_RESOURCE_LIST_NAME and P_RESOURCE_LIST_ID
  - RESOURCE_ALIAS and RESOURCE_LIST_MEMBER_ID
- You can specify values for the parameters PA_TASK_ID or PM_TASK_REFERENCE only when budgeting by tasks, as defined by the budget entry method.
- You can specify values for the parameter PERIOD_NAME only when budgeting by PA or GL period, as defined by the budget entry method.
- If you budget by PA or GL period and do not provide a period name, Oracle Projects uses the budget start date and budget end date to select a valid period name from the database. If Oracle Projects fails to retrieve a valid period name, the API will abort.
- The task level at which you pass budget information should correspond to the level specified in the budget entry method. For example, if the budget entry method specifies that you can

enter a budget only at the lowest task level, then this API should pass only lowest tasks.

- When the budget entry method flags shown in the following table are set to N, do not pass the related parameters.

BEM Flag	*Related Parameter*
COST_QUANTITY_FLAG	QUANTITY
RAW_COST_FLAG	RAW_COST
BURDENED_COST_FLAG	BURDENED_COST
REV_QUANTITY_FLAG	QUANTITY
REVENUE_FLAG	REVENUE

Table 6 – 5 Parameters not to Pass if BEM Flags Are Set to N (Page 1 of 1)

- You can add a budget line only to a budget with a status of Working.
- Since Oracle Projects identifies a budget line by its budget start date, you cannot update the P_BUDGET_START_DATE. You can update only the budget line parameters below:
  - P_DESCRIPTION
  - P_RAW_COST
  - P_BURDENED_COST
  - P_REVENUE
  - P_QUANTITY
- When passed, the parameters marked with an asterisk (*) in the table below are used to identify the budget line.
- Although flexfield parameters appear in the parameter list below, this API does not currently use them to update budget line flexfields.

The following table shows the parameters for UPDATE_BUDGET_LINE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard

Name	Usage	Type	Req?	Description
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID*	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE*	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE*	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_PA_TASK_ID*	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE*	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
P_RESOURCE_ALIAS*	IN	VARCHAR2(30)	No	Alias of a resource
P_RESOURCE_LIST_MEMBER_ID*	IN	NUMBER	No	The identification code of the resource
P_BUDGET_START_DATE*	IN	DATE	No	Start date of budget line
P_BUDGET_END_DATE*	IN	DATE	No	End date of budget line
P_PERIOD_NAME*	IN	VARCHAR2(30)	No	GL or PA period name
P_DESCRIPTION	IN	VARCHAR2(255)		(currently unavailable)
P_RAW_COST	IN	NUMBER	No	Budgeted raw cost amount
P_BURDENED_COST	IN	NUMBER	No	Budgeted burdened cost amount
P_REVENUE	IN	NUMBER	No	Budgeted revenue amount
P_QUANTITY	IN	NUMBER	No	Budgeted quantity
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget line descriptive flexfield

CLEAR_BUDGET

CLEAR_BUDGET is a Load–Execute–Fetch procedure used to clear the global data structures set up during the Initialize step.

EXECUTE_CALCULATE_AMOUNTS

EXECUTE_CALCULATE_AMOUNTS is a Load–Execute–Fetch procedure used to calculate the raw cost, burdened cost, and revenue amounts using existing budget lines for a given project and budget type. For each budget line, this API writes to globals that can be read by the API FETCH_CALCULATE_AMOUNTS.

Business Rules

- Some parameters are used only for the Web–based user interface. For budgets that do not use the Web–based interface, thefollowing types of parameters are not applicable:
  - Plan type and plan code parameters
  - Version type parameters
  - P_TIME_PHASED_CODE
  - P_PLAN_IN_MULTI_CURR_FLAG
  - Currency attributes
  - Flags for raw cost, burdened cost, revenue, quantity planning, create current working version, replace current working version
  - P_USING_RESOURCE_LISTS_FLAG
- Since this API calls the PA_CLIENT_EXTN_BUDGET extension, you must modify the extension to calculate the amounts you want.
- You must pass an uppercase 'Y' for each calculation flag to recalculate the corresponding amount.
- Regardless of its update status, this API returns one row of amounts for each budget line it reads.
- To update the budget lines for a project with the calculated amounts generated from this API, you must set the P_UPDATE_DB_FLAG to an uppercase 'Y'.
- This API returns the total number of budget lines processed in the OUT parameter P_TOT_BUDGET_LINES_CALCULATED. This total determines how many times to call FETCH_CALCULATE_AMOUNTS in a loop.

The following table shows the parameters for EXECUTE_CALCULATE_AMOUNTS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_TOT_BUDGET_LINES_CALCULATED	OUT	NUMBER		Indicates the total number of budget lines calculated and determines how many times to call the API FETCH_CALCULATE_AMOUNTS
P_PM_PRODUCT_CODE	IN	VARCHAR2(25)	Yes	The product code of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_CALC_RAW_COST_YN	IN	VARCHAR2(1)	No	Calculate raw cost (Y or N)
P_CALC_BURDENED_COST_YN	IN	VARCHAR2(1)	No	Calculate burdened cost (Y or N)
P_CALC_REVENUE_YN	IN	VARCHAR2(1)	No	Calculate revenue (Y or N)
P_UPDATE_DB_FLAG	IN	VARCHAR2(1)	No	Update budget line (Y or N)
P_BUDGET_VERSION_ID	IN	NUMBER(15)	No	The system-generated number that uniquely identifies the budget version
P_FIN_PLAN_TYPE_ID	IN	NUMBER	No	The unique identifier of the plan type used to create the plan version
P_FIN_PLAN_TYPE_NAME	IN	VARCHAR2(150)	No	The plan type name. You must supply a valid value for either p_fin_plan_type_id or p_fin_plan_type_name.
P_VERSION_TYPE	IN	VARCHAR2(30)	No	The version type. A value for this parameter is required when the plan type is set up to plan cost and revenue versions separately. In other cases, if the value is not passed, then it is derived automatically.
P_BUDGET_VERSION_NUMBER	IN	NUMBER(15)	No	The version number for the budget

EXECUTE_CREATE_DRAFT_BUDGET

EXECUTE_CREATE_DRAFT_BUDGET is used to create a budget and its budget lines using the data stored in the global tables during the Load process.

Business Rules

- Some parameters are used only for the Web–based user interface. For budgets that do not use the Web–based interface, thefollowing types of parameters are not applicable:
  - Plan type and plan code parameters
  - Version type parameters
  - P_TIME_PHASED_CODE
  - P_PLAN_IN_MULTI_CURR_FLAG
  - Currency attributes
  - Flags for raw cost, burdened cost, revenue, quantity planning, create current working version, replace current working version
  - P_USING_RESOURCE_LISTS_FLAG

The following table shows the parameters for EXECUTE_CREATE_DRAFT_BUDGET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PM_BUDGET_REFERENCE	IN	VARCHAR2(30)	No	The reference code that identifies the budget on the client side
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system

Name	Usage	Type	Req?	Description
P_BUDGET_VERSION_NAME	IN	VARCHAR2	No	The user-defined name for the budget version
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_CHANGE_REASON_CODE	IN	VARCHAR2(30)	No	The reference code that identifies the change reason
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
P_ATTRIBUTE1 through P_ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget descriptive flexfield
P_DESCRIPTION	IN	VARCHAR2(255)	No	Description of the budget
P_ENTRY_METHOD_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget entry method
P_RESOURCE_LIST_NAME	IN	VARCHAR2(60)	No	Name of the resource list
P_RESOURCE_LIST_ID	IN	NUMBER	No	The identification code of the resource list
P_FIN_PLAN_TYPE_ID	IN	NUMBER	No	The unique identifier of the plan type being used for the creation of a plan version
P_FIN_PLAN_TYPE_NAME	IN	VARCHAR2(150)	No	The plan type name. You must supply a valid value for either p_fin_plan_type_id or p_fin_plan_type_name
P_VERSION_TYPE	IN	VARCHAR2(30)	No	The version type. A value for this parameter is required when the plan type is set up to plan cost and revenue versions separately. In other cases, if the value is not passed, then it is derived automatically.
P_FIN_PLAN_LEVEL_CODE	IN	VARCHAR2(30)	No	The planning level for the plan version. Valid values are P (project-level planning), T (top task-level planning), M (mixed-level planning - top and lowest tasks), and L (lowest task-level planning).
P_TIME_PHASED_CODE	IN	VARCHAR2(30)	No	The time phasing option. Valid values are P (planning by PA periods), G (planning by GL periods), R (planning by date range), and N (None-implies that dates are derived from the project or task dates.
P_PLAN_IN_MULTI_CURR_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether the plan version uses multiple transaction currencies
P_PROJFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting cost amounts from the transaction currency to the project functional currency

Name	Usage	Type	Req?	Description
P_PROJFUNC_COST_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting cost amounts from transaction currency to project functional currency
P_PROJFUNC_COST_RATE_DATE	IN	DATE	No	The rate date for converting cost amounts from transaction currency to project functional currency
P_PROJFUNC_REV_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting revenue amounts from the transaction currency to the project functional currency
P_PROJFUNC_REV_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting revenue amounts from transaction currency to project functional currency
P_PROJFUNC_REV_RATE_DATE	IN	DATE	No	The rate date for converting revenue amounts from transaction currency to project functional currency
P_PROJECT_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting cost amounts from the transaction currency to the project currency
P_PROJECT_COST_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting cost amounts from transaction currency to project currency
P_PROJECT_COST_RATE_DATE	IN	DATE	No	The rate date for converting cost amounts from transaction currency to project currency
P_PROJECT_REV_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting revenue amounts from the transaction currency to the project currency
P_PROJECT_REV_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting revenue amounts from transaction currency to project currency
P_PROJECT_REV_RATE_DATE	IN	DATE	No	The rate date for converting revenue amounts from transaction currency to project currency
P_RAW_COST_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether raw cost can be planned for the plan version
P_BURDENED_COST_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether burdened cost can be planned for the plan version
P_REVENUE_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether revenue can be planned for the plan version

Name	Usage	Type	Req?	Description
P_COST_QTY_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether quantity can be planned for the plan version of version type cost
P_REVENUE_QTY_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether quantity can be planned for the plan version of version type revenue
P_ALL_QTY_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether quantity can be planned when cost and revenue are planned together in a single plan version
P_CREATE_NEW_CURR_WORKING_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether a current working version should be created. This parameter is required only if the budget uses the Web-based user interface.
P_REPLACE_CURRENT_WORKING_FLAG	IN	VARCHAR2(1)	No	Flag indicating whether the current working version should be deleted and the newly created version marked as the Current Working version. This parameter is required only if the budget uses the Web-based user interface.
P_USING_RESOURCE_LISTS_FLAG	IN	VARCHAR2(1)	No	When a plan amount is not classified using a resource list, then N (no) must be entered as the parameter value. This parameter is required only if the budget uses the Web-based user interface.

EXECUTE_UPDATE_BUDGET

EXECUTE_UPDATE_BUDGET is a Load–Execute–Fetch procedure used to update a budget and its budget lines using the data stored in the global tables during the Load process.

Business Rules

Note: This API does not support the Web–based budget user interface.

The following table shows the parameters for EXECUTE_UPDATE_BUDGET.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_MSG_COUNT	OUT	NUMBER		API standard
P_MSG_DATA	OUT	VARCHAR2(2000)		API standard
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PM_PRODUCT_CODE	IN	VARCHAR2(10)	Yes	The product code of the vendor of the external system
P_PA_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_BUDGET_TYPE_CODE	IN	VARCHAR2(30)	Yes	The reference code that identifies the budget type
P_CHANGE_REASON_CODE	IN	VARCHAR2(30)	No	The reference code that identifies the change reason
P_DESCRIPTION	IN	VARCHAR2(255)	No	Description of the budget

FETCH_BUDGET_LINE

FETCH_BUDGET_LINE is a Load–Execute–Fetch procedure used to retrieve the return status returned during the creation of a budget line from a global PL/SQL table.

The following table shows the parameters for FETCH_BUDGET_LINE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)		API standard (default = 'F')
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_LINE_INDEX	IN	NUMBER	Yes	Pointer to specific budget line
P_LINE_RETURN_STATUS	OUT	VARCHAR2(1)		Return status for specific line

FETCH_CALCULATE_AMOUNTS

FETCH_CALCULATE_AMOUNTS is a Load–Execute–Fetch procedure used to get the raw cost, burdened cost, and revenue amounts by budget line from global records updated by the API EXECUTE_CALCULATE_AMOUNTS.

Business Rule

- Call this API in a loop for each calculated budget line using the API EXECUTE_CALCULATE_AMOUNTS. The value the API EXECUTE_CALCULATE_AMOUNTS returns for P_TOT_BUDGET_LINES_CALCULATED determines how many times to call this API.

The following table shows the parameters for FETCH_CALCULATE_AMOUNTS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_LINE_INDEX	OUT	NUMBER		Pointer to specific budget line
P_RETURN_STATUS	OUT	VARCHAR2(1)		API standard
P_PA_TASK_ID	OUT	NUMBER		The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE	OUT	VARCHAR2(30)		The reference code that uniquely identifies the task in the external system
P_BUDGET_START_DATE	OUT	DATE		Start date of budget line
P_BUDGET_END_DATE	OUT	DATE		End date of budget line
P_PERIOD_NAME	OUT	VARCHAR2(20)		PA or GL period name
P_RESOURCE_LIST_MEMBER_ID	OUT	NUMBER		The identification code of the resource
P_QUANTITY	OUT	NUMBER		The quantity entered into the budget line
P_RESOURCE_ALIAS	OUT	VARCHAR2(30)		Alias of resource
P_CALCULATED_RAW_COST	OUT	NUMBER		Calculated raw cost
P_CALCULATED_BURDENED_COST	OUT	NUMBER		Calculated burdened cost
P_CALCULATED_REVENUE	OUT	NUMBER		Calculated revenue
P_LINE_RETURN_STATUS	OUT	VARCHAR2(1)		Return status for a specific line

INIT_BUDGET

INIT_BUDGET is a Load–Execute–Fetch procedure used to set up the global data structures that other Load–Execute–Fetch procedures use to create a new or update an existing draft budget in Oracle Projects.

INIT_CALCULATE_AMOUNTS

INIT_CALCULATE_AMOUNTS is a Load–Execute–Fetch procedure used to set up the global data structures used by the Load–Execute–Fetch API CALCULATE_AMOUNTS.

LOAD_BUDGET_LINE

LOAD_BUDGET_LINE is a Load–Execute–Fetch procedure used to load a budget line to a global PL/SQL table.

Business Rules

- Some parameters are used only for the Web–based user interface. For budgets that do not use the Web–based interface, thefollowing types of parameters are not applicable:
  - P_TXN_CURRENCY_CODE
  - Currency attributes
  - P_CHANGE_REASON_CODE

The following table shows the parameters for LOAD_BUDGET_LINE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API standard
P_COMMIT	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	API standard (default = 'F')
P_RETURN_STATUS	OUT	VARCHAR2(1)	No	API standard
P_PA_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	Alias of a resource

Name	Usage	Type	Req?	Description
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	No	The identification code of the resource
P_BUDGET_START_DATE	IN	DATE	No	Start date of budget line
P_BUDGET_END_DATE	IN	DATE	No	End date of budget line
P_PERIOD_NAME	IN	VARCHAR2(30)	No	PA or GL period name
P_DESCRIPTION	IN	VARCHAR2(255)	No	Description of the budget
P_RAW_COST	IN	NUMBER	No	Budgeted raw cost amount
P_BURDENED_COST	IN	NUMBER	No	Budgeted burdened cost amount
P_REVENUE	IN	NUMBER	No	Budgeted revenue amount
P_QUANTITY	IN	NUMBER	No	Budgeted quantity
PM_PRODUCT_CODE	IN	VARCHAR2(30)	No	The product code of the vendor of the external system
PM_BUDGET_LINE_REFERENCE	IN	VARCHAR2(30)	No	The reference code that identifies the budget line on the client side
P_TXN_CURRENCY_CODE	IN	VARCHAR2(30)	Yes	The transaction currency code in which the budget line is being planned. This parameter is required for budgets that use the Web-based user interface.
P_PROJFUNC_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting cost amounts from the transaction currency to the project functional currency
P_PROJFUNC_COST_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting cost amounts from transaction currency to project functional currency
P_PROJFUNC_COST_RATE_DATE	IN	DATE	No	The rate date for converting cost amounts from transaction currency to project functional currency
P_PROJFUNC_REV_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting revenue amounts from the transaction currency to the project functional currency
P_PROJFUNC_REV_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting revenue amounts from transaction currency to project functional currency
P_PROJFUNC_REV_RATE_DATE	IN	DATE	No	The rate date for converting revenue amounts from transaction currency to project functional currency
P_PROJECT_COST_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting cost amounts from the transaction currency to the project currency

Name	Usage	Type	Req?	Description
P_PROJECT_COST_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting cost amounts from transaction currency to project currency
P_PROJECT_COST_RATE_DATE	IN	DATE	No	The rate date for converting cost amounts from transaction currency to project currency
P_PROJECT_REV_RATE_TYPE	IN	VARCHAR2(30)	No	The rate type for converting revenue amounts from the transaction currency to the project currency
P_PROJECT_REV_RATE_DATE_TYP	IN	VARCHAR2(30)	No	The rate date type for converting revenue amounts from transaction currency to project currency
P_PROJECT_REV_RATE_DATE	IN	DATE	No	The rate date for converting revenue amounts from transaction currency to project currency
P_CHANGE_REASON_CODE	IN	VARCHAR2(30)	No	The reason for change when entering or modifying the amounts in a plan version.
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)	No	Used by descriptive flexfields
ATTRIBUTE1 through ATTRIBUTE15	IN	VARCHAR2(150)	No	Budget line descriptive flexfield

Using Budget APIs

The following example describes how to create an interface between Oracle Projects and the budget and budget line information in your external system. Depending on your company's business needs, your own implementation of budget APIs may be more or less complex than the scenario shown here.

As you work through this example, you may want to refer to information elsewhere in this manual:

- For a detailed description of the budget APIs, see Budget APIs: page 6 – 2.
- Most of the Oracle Projects APIs use a standard set of input and output parameters. See Standard API Parameters: page 2 – 19.
- For an example of PL / SQL code that creates a budget using Load–Execute–Fetch APIs, see Creating a Budget Using the Load–Execute–Fetch APIs: page 6 – 48.
- For an example of PL / SQL code that creates a budget using APIs that use composite datatypes, see Creating a Budget Using a Composite Datatype API: page 6 – 53.

Step 1 Connect to an Oracle database

To ensure that proper security is enforced while accessing Oracle Projects data, follow the steps in Security Requirements: page 2 – 9.

Step 2 Get the budget data

Before you send budget lines to the Oracle Projects database, you must first make some decisions that affect how the budget and budget lines are linked to other Oracle Projects data. This section provides sample PL / SQL select statements upon which you can model your own. The following pages describe the relationship between the selected values and budget or budget line information. Understanding this relationship helps you to determine which parameter values to pass to the budget and budget line APIs.

▶ **Select the budget type:**

Select a valid budget type. Oracle Projects predefines the budget types shown in the following table:

Budget Type Code	*Budget Type*
AC	Approved Cost Budget
AR	Approved Revenue Budget
FC	Forecast Cost Budget
FR	Forecast Revenue Budget

Table 6 – 6 Budget types predefined by Oracle Projects (Page 1 of 1)

The following PL/SQL statement retrieves the budget type information:

```
SELECT code,
       name
FROM   pa_budget_types_v
```

The selected value &CODE is related to the budget parameter P_BUDGET_TYPE_CODE.

Because cost and revenue budgets can contain different budget amounts, you must retrieve the budget amount code for the budget type. The following PL/SQL statement retrieves the appropriate budget amount code:

```
SELECT budget_amount_code
FROM pa_budget_types
WHERE budget_type_code = &code
```

The statement returns C if you have chosen a cost budget, and R if you have chosen a revenue budget. The following table illustrates the amounts each budget type can hold and their relation to the parameters of LOAD_BUDGET_LINE:

Amount	*LOAD_BUDGET_LINE Parameter*
Raw Cost	P_RAW_COST
Burdened Cost	P_BURDENED_COST
Cost Quantity	P_QUANTITY

Table 6 – 7 How budget amounts relate to API parameters (Page 1 of 2)

Amount	*LOAD_BUDGET_LINE Parameter*
Revenue	P_REVENUE
Revenue Quantity	P_QUANTITY

Table 6 – 7 How budget amounts relate to API parameters (Page 2 of 2)

▶ **Select the budget entry method:**

Oracle Projects predefines the budget entry methods shown in the following table:

Budget Entry Method Code	Budget Entry Method
PA_LOWEST_TASK_BY_PA_PERIOD	By lowest tasks and PA period, categorized by resource
PA_LOWEST_TASK_BY_GL_PERIOD	By lowest tasks and GL period, categorized by resource
PA_LOWEST_TASK_BY_DATE_RANGE	By lowest tasks and date range, categorized by resource

Table 6 – 8 Budget entry methods predefined by Oracle Projects (Page 1 of 1)

The following PL/SQL statement retrieves the budget entry method:

```
SELECT code
, name
, categorization_code
, entry_level_code
, entry_level_name
, time_phased_type_code
, time_phased_type_name
FROM pa_budget_entry_methods_v
```

The selected value CODE is related to the budget parameter P_ENTRY_METHOD_CODE.

You can use the other selected values later to retrieve other budget–related data from Oracle Projects. Possible values for other budget–related fields include:

- For CATEGORIZATION_CODE
  R Categorized by resource
  N Not categorized
- For ENTRY_LEVEL_CODE
  P Budgeting at the project level
  T Budgeting at the top task level
  L Budgeting at the lowest task level
  M Budgeting at both top and lowest task (mixed) level
- For &TIME_PHASED_TYPE_CODE
  P Budget lines by PA periods
  G Budget lines by GL periods
  R Budget lines by date ranges
  N Budget lines not time–phased

▶ **Select a resource list:**

If you select a budget entry method that is categorized by resources, you must select a resource list for the budget. The following PL/SQL statement retrieves the resource list information:

```
SELECT resource_list_id
, resource_list_name
, description
FROM pa_qry_resource_lists_v
```

The following table illustrates the relationship between certain selected values and budget parameters. Pass only one of the two values:

Selected Value	*Budget Parameter*
&RESOURCE_LIST_ID	P_RESOURCE_LIST_ID
RESOURCE_LIST_NAME	P_RESOURCE_LIST_NAME

Table 6 – 9 How selected values relate to budget parameters (Page 1 of 1)

▶ **Select other budget–related parameters:**

The parameter P_DESCRIPTION holds the description for a budget. Use the view PA_BUDGET_CHANGE_REASON_V to pass an explanation for any changes made to the budget. The following PL/SQL statement retrieves the reason for the budget change:

```
SELECT code,
       name
FROM pa_budget_change_reason_v
```

The following table illustrates the relationship between certain selected values and budget parameters:

Selected Value	*Budget Parameter*
CODE	P_CHANGE_REASON_CODE
DESCRIPTION	P_DESCRIPTION

Table 6 – 10 How selected values relate to budget parameters (Page 1 of 1)

Step 3 Get budget line data

The choices you made for your budget data strongly affect your budget line data. These effects are described on the following pages.

▶ **Select amount fields:**

As shown above, cost budgets can contain raw cost, burdened cost, and cost quantity amounts, while revenue budgets can contain only revenue and revenue quantity amounts.

▶ **Select tasks:**

Depending on the budget entry level, the budget line should include the appropriate TASK_ID or TASK_REFERENCE. With project–level budgeting, you do not pass task–related parameters. With task–level budgeting (top, lowest, or mixed), you can use the following PL / SQL statements to retrieve valid task values:

- Budget at the top task level

```
SELECT task_id
, pm_task_reference
, task_number
, task_name
FROM pa_tasks
WHERE project_id = &project_id
AND parent_task_id IS NULL
```

- Budget at the lowest task level

```
SELECT task_id
, pm_task_reference
, task_number
, task_name
FROM pa_tasks tasks1
WHERE     tasks1.project_id = &project_id
and not EXISTS (select  NULL
    from pa_tasks tasks2
    where tasks1.project_id = tasks2.project_id
    and tasks1.task_id = tasks2.parent_task_id)
```

- Budget at the top and lowest task levels

```
SELECT task_id
, pm_task_reference
, task_number
, task_name
, decode(nvl(parent_task_id,'1'),1,'Y','N') TOP_TASK
FROM pa_tasks tasks1
WHERE tasks1.project_id = &project_id
and not EXISTS (select  NULL
    from pa_tasks tasks2
    where  tasks1.project_id = tasks2.project_id
    and tasks1.task_id = tasks2.parent_task_id)
or  (tasks1.parent_task_id IS NULL
    and tasks1.project_id = &project_id )
```

The following table illustrates the relationship between certain selected values and budget line parameters. Pass only one of the two values:

Selected Value	***Budget Line Parameter***
TASK_ID	P_PA_TASK_ID
PM_TASK_REFERENCE	P_PM_TASK_REFERENCE

Table 6 – 11 How selected values relate to budget line parameters (Page 1 of 1)

▶ **Select resource list members (resources):**

If your budget entry method is categorized by resources and you have selected a resource list, the budget line should include the individual resources associated with the resource list. You can use the following PL / SQL statement to retrieve the resource list member information:

```
SELECT resource_list_member_id
, alias
, employee_first_name
, employee_last_name
FROM pa_query_res_list_members_v
WHERE resource_list_id = &resource_list_id
```

The following table illustrates the relationship between certain of the selected values and budget line parameters. Pass only one of the two values:

Selected Value	*Budget Line Parameter*
RESOURCE_LIST_MEMBER_ID	P_RESOURCE_LIST_ MEMBER_ID
ALIAS	P_RESOURCE_ALIAS

Table 6 – 12 How selected values relate to budget line parameters (Page 1 of 1)

▶ **Select periods:**

How the budget entry method is time–phased affects which budget line parameters accept passed values, as shown in the following table:

Time–Phased By	**Parameters That Accept Values (START_DATE, END_DATE, and PERIOD_NAME)**
No Time–Phasing	None
Date Ranges	START_DATE and END_DATE
PA or GL Period	START_DATE and END_DATE or PERIOD_NAME

Table 6 – 13 How budget entry methods affect budget line parameters (Page 1 of 1)

When using time–phased budgeting, you can use the following PL/SQL statements to retrieve the appropriate date information:

- Period name

```
SELECT period_name
FROM pa_budget_periods_v
WHERE period_type_code = &time_phased_type_code
```

- Begin and end dates

```
SELECT period_start_date
, period_end_date
FROM pa_budget_periods_v
WHERE period_type_code = &time_phased_type_code
```

The following table illustrates the relationship between certain selected values and budget line parameters. You can pass a value for either the PERIOD_NAME or both the PERIOD_START_DATE and PERIOD_END_DATE:

Selected Value	*Budget Line Parameter*
PERIOD_NAME	P_PERIOD_NAME
PERIOD_START_DATE	P_BUDGET_START_DATE
PERIOD_END_DATE	P_BUDGET_END_DATE

Table 6 – 14 How selected values relate to budget line parameters (Page 1 of 1)

▶ **Select descriptions:**

You do not need to pass a description for budget lines.

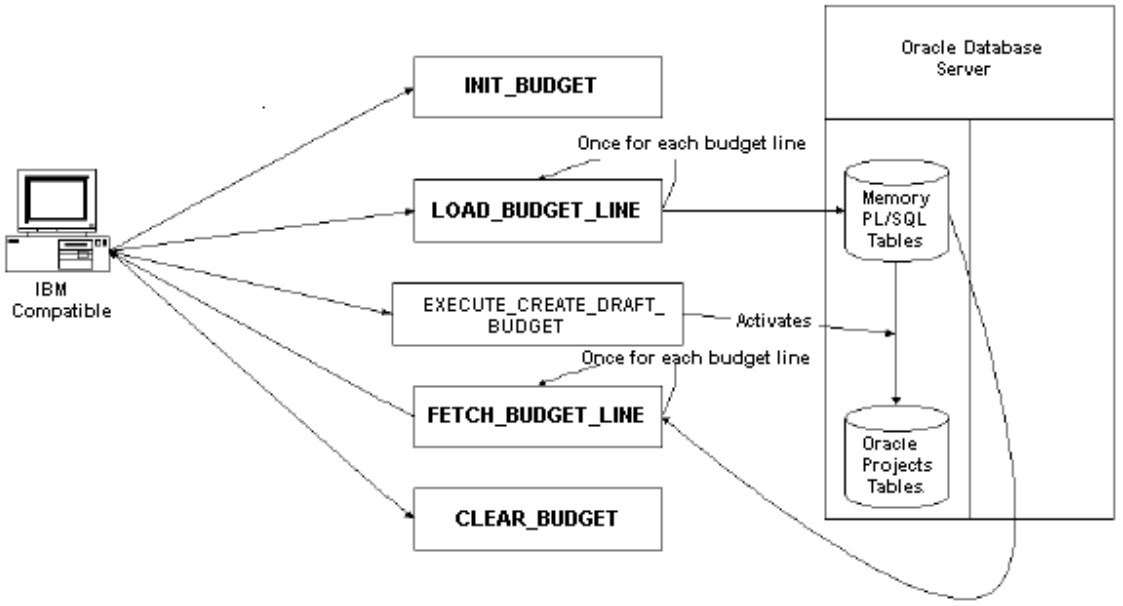
Step 4 Interface budget information to the server

If your external system supports composite datatype parameters, such as Oracle PL/SQL Version 2.3 or higher, you can call the CREATE_DRAFT_BUDGET and UPDATE_BUDGET APIs directly.

Not all external systems can call the APIs that use composite datatypes. Systems that do not support composite datatypes must call the supplementary Load–Execute–Fetch APIs. The Load–Execute–Fetch procedures include Initialize, Load, Execute, Fetch, and Clear categories. For more information, see API Procedures: page 2 – 31.

The following figure illustrates the flow of the Load–Execute–Fetch procedures for budget APIs. The process first calls the API INIT_BUDGET, which resets the server–side global PL/SQL tables that temporarily store the budget and budget line data. Once you set up these tables, use LOAD_BUDGET_LINE to move the budget and budget line data to the Oracle Projects database.

Figure 6 – 1 Load–Execute–Fetch Procedures for Budget APIs



Step 5 Start the server–side process

After the Load procedure successfully moves budget and budget line data to the Oracle Projects database, call the procedure API EXECUTE_CREATE_DRAFT_BUDGET to process the budget and budget line data in the global PL/SQL tables.

Step 6 Retrieve error messages

Each Oracle Projects API includes standard output parameters:

- P_RETURN_STATUS shows if the API was executed successfully.
- P_MSG_COUNT shows the number of errors detected during the execution of the API.

If the API detects one error, the API returns the error message text. If the API detects multiple errors, use GET_MESSAGES to retrieve the error messages. See GET_MESSAGES: page 2 – 24.

If the error relates to a budget line, use FETCH_BUDGET_LINE to identify the line causing the error. The API parameter P_LINE_RETURN_STATUS identifies the line by returning either E (business rule violation) or U (unexpected error) for that line. (For more information about the return status, see Standard API Parameters: page 2 – 19. If you use FETCH_BUDGET_LINE for any other reason, it returns the error NO_DATA_FOUND.

Step 7 Finish the Load–Execute–Fetch process

After executing the Fetch procedures and retrieving any error messages, finish the Load–Execute–Fetch process by calling the API CLEAR_BUDGET and either saving or rolling back your changes to the database.

Creating a Budget Using the Load–Execute–Fetch APIs

The following sample PL/SQL code is a sample of a script you can use to create a budget using the Load–Execute–Fetch APIs.

The Load–Execute–Fetch APIs use parameters with standard datatypes (VARCHAR2, NUMBER, and DATE). They do not use composite datatypes. If you create budgets using tools or products that support composite datatypes, see Creating a Budget Using a Composite Datatype API: page 6 – 53.

```
DECLARE
  --variables needed for API standard parameters
  l_api_version_number        NUMBER :=1.0;
  l_commit                    VARCHAR2(1):= 'F';
  l_return_status       VARCHAR2(1);
  l_init_msg_list       VARCHAR2(1);
  l_msg_count                 NUMBER;
  l_msg_data                  VARCHAR2(2000);
  l_data                      VARCHAR2(2000);
  l_msg_entity                VARCHAR2(100);
  l_msg_entity_index          NUMBER;
  l_msg_index                 NUMBER;
  l_msg_index_out       NUMBER;
  l_encoded                   VARCHAR2(1);
  i                           NUMBER;
  a                           NUMBER;
  --variables needed for Oracle Project specific parameters
  l_pm_product_code           VARCHAR2(10);
  l_pa_project_id       NUMBER;
  l_pm_project_reference      VARCHAR2(25);
  l_budget_type_code          VARCHAR2(30);
  l_change_reason_code        VARCHAR2(30);
  l_description               VARCHAR2(255);
  l_entry_method_code         VARCHAR2(30);
  l_resource_list_name        VARCHAR2(60);
  l_resource_list_id          NUMBER;
  l_budget_lines_in           pa_budget_pub.budget_line_in_t
bl_type;
  l_budget_lines_in_rec       pa_budget_pub.budget_line_in_r
ec_type;
  l_budget_lines_out          pa_budget_pub.budget_line_out_
tbl_type;
  l_line_index                NUMBER;
  l_line_return_status        VARCHAR2(1);
```

```
  API_ERROR                    EXCEPTION;
BEGIN
--PRODUCT RELATED DATA
  l_pm_product_code :='SOMETHING';
--BUDGET DATA
--l_pa_project_id:=  1138;
l_pm_project_reference := 'PROJECT_NAME';
l_budget_type_code := 'AC';
l_change_reason_code := 'ESTIMATING ERROR';
l_description := 'New description -> 2';
l_entry_method_code := 'PA_LOWEST_TASK_BY_DATE_RANGE';
l_resource_list_id := 1014;
--BUDGET LINES DATA
a := 5;
FOR i IN 1..a LOOP
if i = 1 THEN
l_budget_lines_in_rec.pa_task_id := 2440;
l_budget_lines_in_rec.resource_list_member_id := 1401;
elsif i = 2 THEN
l_budget_lines_in_rec.resource_list_member_id := 1402;
l_budget_lines_in_rec.pa_task_id := 2443;
elsif i = 3 THEN
l_budget_lines_in_rec.resource_list_member_id := 1404;
l_budget_lines_in_rec.pa_task_id := 2446;
elsif i = 4 THEN
l_budget_lines_in_rec.resource_list_member_id := 1407;
l_budget_lines_in_rec.pa_task_id := 2449;
elsif i = 5 THEN
l_budget_lines_in_rec.resource_list_member_id := 1408;
l_budget_lines_in_rec.pa_task_id := 2452;
end if;
l_budget_lines_in_rec.quantity :=93;
l_budget_lines_in_rec.budget_start_date :=  '05-MAY-95';
l_budget_lines_in_rec.budget_end_date :=  '09-MAY-95';
l_budget_lines_in_rec.raw_cost :=300;
l_budget_lines_in(i) := l_budget_lines_in_rec;
END LOOP;
-----------------------
--INIT_BUDGET
pa_budget_pub.init_budget;
-----------------------
--LOAD_BUDGET_LINE
FOR i IN 1..a LOOP
```

```
pa_budget_pub.load_budget_line( p_api_version_number =>
l_api_version_number
                ,p_return_status => l_return_status
                ,p_pa_task_id =>
l_budget_lines_in(i).pa_task_id
                ,p_pm_task_reference =>
l_budget_lines_in(i).pm_task_reference
                ,p_resource_alias   =>
l_budget_lines_in(i).resource_alias
                ,p_resource_list_member_id =>

l_budget_lines_in(i).resource_list_member_id
                ,p_budget_start_date =>
l_budget_lines_in(i).budget_start_date
                ,p_budget_end_date =>
l_budget_lines_in(i).budget_end_date
                ,p_period_name =>
l_budget_lines_in(i).period_name
                ,p_description =>
l_budget_lines_in(i).description
                ,p_raw_cost =>
l_budget_lines_in(i).raw_cost
                ,p_burdened_cost =>
l_budget_lines_in(i).burdened_cost
                ,p_revenue => l_budget_lines_in(i).revenue
                ,p_quantity =>
l_budget_lines_in(i).quantity );
END LOOP;
IF l_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;
-----------------------
--EXECUTE_CREATE_DRAFT_BUDGET
pa_budget_pub.execute_create_draft_budget
( p_api_version_number => l_api_version_number
 ,p_msg_count => l_msg_count
 ,p_msg_data => l_msg_data
 ,p_return_status => l_return_status
 ,p_pm_product_code => l_pm_product_code
 ,p_pa_project_id => l_pa_project_id
 ,p_pm_project_reference => l_pm_project_reference
 ,p_budget_type_code => l_budget_type_code
 ,p_change_reason_code => l_change_reason_code
```

```
 ,p_description => l_description
 ,p_entry_method_code => l_entry_method_code
 ,p_resource_list_name => l_resource_list_name
 ,p_resource_list_id => l_resource_list_id    );
IF l_return_status != 'S'
THEN
   null; --RAISE API_ERROR;
END IF;
-----------------------
--FETCH_LINE
FOR l_line_index in
1..PA_BUDGET_PUB.G_budget_lines_tbl_count LOOP
pa_budget_pub.fetch_budget_line( p_api_version_number =>
l_api_version_number
                ,p_return_status => l_return_status
                ,p_line_index => l_line_index
                ,p_line_return_status =>
l_line_return_status);
IF l_return_status != 'S'
OR l_line_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;
END LOOP;
-----------------------
--CLEAR_BUDGET
pa_budget_pub.clear_budget;
IF l_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;
-----------------------
--HANDLE EXCEPTIONS
EXCEPTION
WHEN API_ERROR THEN
    for i in 1..l_msg_count loop
        pa_interface_utils_pub.get_messages (
                        p_msg_data => l_msg_data
                    ,p_data => l_data
                       ,p_msg_count => l_msg_count
                ,p_msg_index_out => l_msg_index_out );
        dbms_output.put_line ('error mesg '||l_data);
        dbms_output.put_line ('error mesg '||l_msg_data);
```

```
    end loop;
WHEN OTHERS THEN
    for i in 1..l_msg_count loop
        pa_interface_utils_pub.get_messages (
                         p_msg_data => l_msg_data
                      ,p_data => l_data
                        ,p_msg_count => l_msg_count
                  ,p_msg_index_out => l_msg_index_out );
        dbms_output.put_line ('error mesg '||l_data);

    end loop;
END;
/
```

Creating a Budget Using a Composite Datatype API

The following sample PL/SQL code is a script that creates a budget using the API CREATE_DRAFT_BUDGET, which uses composite datatypes. If you create budgets using tools or products that do not support composite datatypes, see Creating a Budget Using the Load–Execute–Fetch APIs: page 6 – 48.

```
DECLARE
--variables needed for API standard parameters
l_api_version_number    NUMBER :=1.0;
l_commit                VARCHAR2(1):= 'F';
l_return_status         VARCHAR2(1);
l_init_msg_list         VARCHAR2(1);
l_msg_count             NUMBER;
l_msg_data              VARCHAR2(2000);
l_data                  VARCHAR2(2000);
l_msg_entity            VARCHAR2(100);
l_msg_entity_index      NUMBER;
l_msg_index             NUMBER;
l_msg_index_out         NUMBER;
l_encoded               VARCHAR2(1);
i                       NUMBER;
a                       NUMBER;
--variables needed for Oracle Projects-specific parameters
l_pm_product_code VARCHAR2(10);
l_pa_project_id         NUMBER;
l_pm_project_reference  VARCHAR2(25);
l_budget_type_code      VARCHAR2(30);
l_version_name          VARCHAR2(30);
l_change_reason_code    VARCHAR2(30);
l_description           VARCHAR2(255);
l_entry_method_code     VARCHAR2(30);
l_resource_list_name    VARCHAR2(60);
l_resource_list_id      NUMBER;
l_budget_lines_in pa_budget_pub.budget_line_in_tbl_type;
l_budget_lines_in_rec   pa_budget_pub.budget_line_in_rec_typ
e;
l_budget_lines_out      pa_budget_pub.budget_line_out_tbl_ty
pe;
l_line_index            NUMBER;
l_line_return_status    VARCHAR2(1);

API_ERROR               EXCEPTION;
```

```
BEGIN
--PRODUCT RELATED DATA
l_pm_product_code :='SOMETHING';
--BUDGET DATA
l_pm_project_reference := 'PROJECT_NAME';
l_budget_type_code := 'AC'; '--AR'; --
l_change_reason_code := 'ESTIMATING ERROR';
l_description := 'New description 2';
l_version_name := 'New version ';
l_entry_method_code := 'PA_LOWEST_TASK_BY_DATE_RANGE';
l_resource_list_id := 1014;
--BUDGET LINES DATA
a := 5;
FOR i IN 1..a LOOP
if i = 1 THEN
l_budget_lines_in_rec.pa_task_id := 2440;
l_budget_lines_in_rec.resource_list_member_id := 1401;
elsif i = 2 THEN
l_budget_lines_in_rec.resource_list_member_id := 1402;
l_budget_lines_in_rec.pa_task_id := 2443;
elsif i = 3 THEN
l_budget_lines_in_rec.resource_list_member_id := 1404;
l_budget_lines_in_rec.pa_task_id := 2446;
elsif i = 4 THEN
l_budget_lines_in_rec.resource_list_member_id := 1407;
l_budget_lines_in_rec.pa_task_id := 2449;
elsif i = 5 THEN
l_budget_lines_in_rec.resource_list_member_id := 1408;
l_budget_lines_in_rec.pa_task_id := 2452;
end if;
l_budget_lines_in_rec.quantity :=93;
l_budget_lines_in_rec.budget_start_date := '05-MAY-95';
l_budget_lines_in_rec.budget_end_date := '09-MAY-95';
l_budget_lines_in_rec.raw_cost :=300;
l_budget_lines_in(i) := l_budget_lines_in_rec;
END LOOP;
-----------------------
--INIT_BUDGET
pa_budget_pub.init_budget;
-----------------------
--CREATE_DRAFT_BUDGET
pa_budget_pub.create_draft_budget
( p_api_version_number => l_api_version_number
,p_msg_count => l_msg_count
```

```
,p_msg_data => l_msg_data
,p_return_status => l_return_status
,p_pm_product_code => l_pm_product_code
,p_pa_project_id => l_pa_project_id
,p_pm_project_reference => l_pm_project_reference
,p_budget_type_code => l_budget_type_code
,p_change_reason_code => l_change_reason_code
,p_budget_version_name => l_version_name
,p_description => l_description
,p_entry_method_code => l_entry_method_code
,p_resource_list_name => l_resource_list_name
,p_resource_list_id => l_resource_list_id
,p_budget_lines_in => l_budget_lines_in
,p_budget_lines_out => l_budget_lines_out );
IF l_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;
-----------------------
--CLEAR_BUDGET

pa_budget_pub.clear_budget;
IF l_return_status != 'S'
THEN
   RAISE API_ERROR;
END IF;
-----------------------
--HANDLE EXCEPTIONS
EXCEPTION
WHEN API_ERROR THEN
   for i in 1..l_msg_count loop
     pa_interface_utils_pub.get_messages (
          p_msg_data => l_msg_data
          ,p_data => l_data
          ,p_msg_count => l_msg_count
          ,p_msg_index_out => l_msg_index_out );
          dbms_output.put_line ('error mesg '||l_data);
   end loop;
WHEN OTHERS THEN
   for i in 1..l_msg_count loop
     pa_interface_utils_pub.get_messages (
          p_msg_data => l_msg_data
          ,p_data => l_data
          ,p_msg_count => l_msg_count
```

```
            ,p_msg_index_out => l_msg_index_out );
            dbms_output.put_line ('error mesg '||l_data);
    end loop;
END;
/
```

Status APIs

Use your external system to calculate and monitor the progress of your project in terms of earned value and percentage complete. Then use the status APIs to report project status inquiry (and billing, if required) to Oracle Projects.

Using the status views described in this section, you can display actual and budgeted amounts in various formats:

- GL period
- PA period
- Work breakdown structure
- Resource
- Burden components

Overview of Status API Views

At the resource level, labor hours (not quantities) are summarized for resources that are tracked as labor. To determine if a resource tracks labor hours but not quantities, join the RESOURCE_LIST_MEMBER_ID to the PA_RESOURCE_LIST_V for the TRACK_AS_LABOR_FLAG column. If TRACK_AS_LABOR_FLAG is Y, the column tracks only labor hours for the resource. Otherwise, quantities are summarized.

For higher–level project and task–level views, the labor hour and quantity summarization rules mentioned above also apply. For example, a project–level labor resource with a TRACK_AS_LABOR_FLAG of Y may show summarized inception–to–date costs, revenues, budgets, and labor hours, but not quantities.

This method of tracking labor hours and quantities has its roots in the way *predefined resources* are summarized in Oracle Projects. All resources in Oracle Projects can be defined as a combination of one or more predefined resources. Predefined resources have three summarization attributes:

- Unit of measure
- Track as labor
- Roll–up actual quantity

The values for the summarization attributes are hard–coded in a view that is used for mapping actuals to resources. The client can change the values by changing the view. The logic of the view is outlined in the two following tables. The following table shows the logic of the view as it relates to predefined resource types.

Predefined Resource Type	Track as Labor	Unit of Measure	Rollup Actual Quantity
Employee	Yes	Hours	No
Job	Yes	Hours	No
Organization	Yes	Hours	No
Expenditure Type	Depends on the expenditure type attribute with track as labor	Unit of measure specified for the expenditure type	Yes
Event Type	No	blank	No
Supplier	No	blank	No
Expenditure Category	Depends whether the expenditure category includes a labor expenditure type	Depends whether the resource type is tracked as labor	No
Revenue Category	Depends whether the revenue category includes a labor expenditure type	Depends whether the resource type is tracked as labor	No

Table 6 – 15 View logic as it relates to predefined resource types (Page 1 of 1)

The following table shows the logic of the view as it relates to predefined resources.

Predefined Resource	Track as Labor	Unit of Measure	Rollup Actual Quantity
Uncategorized	Yes	Hours	No
Unclassified	No	blank	No

Table 6 – 16 View logic as it rleates to predefined resources (Page 1 of 1)

By defining the uncategorized resource as tracking labor, the client can budget labor hours when entering the uncategorized budget.

To facilitate conditional labor hour and quantity queries on resources, TRACK_AS_LABOR_FLAG is also maintained in the resource member list table. You can query TRACK_AS_LABOR_FLAG column via the PA_RESOURCE_LIST_V view.

To write a select statement on a project–level resource view, you can write a SQL statement similar to the following to conditionally return either labor hours or quantities by resource:

```
SELECT rl.resource_alias List
   , decode(rl.resource_track_as_labor_flag,'Y',
ara.actuals_labor_hours_itd,
   'N', ara.actuals_quantity_itd, 0) Units
   , ara.actuals_labor_hours_itd Hours
   , ara.actuals_quantity_itd Qty
FROM  pa_resource_list_v rl
   , pa_accum_rsrc_act_v  ara
WHERE  ara.resource_list_member_id =
rl.resource_list_member_id
AND rl.resource_list_id = 1000
AND ara.project_id = 1043
AND ara.task_id = 0
```

Although the resource list identification code, project identification code, and retrieved data vary by database, the select statement above should return values similar to those shown in the following table:

List	Units	Hours	Quantity
Labor	406	406	0
Senior.Consultant	40	40	0
Principal.Consultant	122	122	0
Senior.Engineer	164	164	0
Principal.Engineer	80	80	0
Travel	4372	0	4372
Air Travel	3762	0	3762
Personal Auto Use	105	0	105

Table 6 – 17 Example Values Returned by the Select Statement (Page 1 of 2)

List	Units	Hours	Quantity
In–House Recoverables	222	0	222
Computer Services	62	0	62
Automobile Rental	50	0	50
Meals	125	0	125
Other Asset	160	0	160
Other Expenses	225	0	225
Lodging	330	0	330
Other Expenses	225	0	225

Table 6 – 17 Example Values Returned by the Select Statement (Page 2 of 2)

List of Status API Views

The following table lists the views that provide parameter data for the status APIs. For detailed description of the views, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

View	Description
PA_ACCUM_CMT_TXNS_V	Retrieves project–, task–, and resource–related commitments. These commitments include line attributes such as commitment number, dates, expenditure type, and expenditure organization. This view retrieves three major sets of project–related commitments: project level commitments (TASK_ID and RESOURCE_LIST_MEMBER_ID are zero), project–task level commitments (RESOURCE_LIST_MEMBER_ID is zero), and project–task–resource level commitments.
PA_ACCUM_RSRC_ACT_V	Returns current project– and task–level resource actual cost and revenue summary amounts by the following periods: inception–to–date, year–to–date, prior period, and period–to–date
PA_ACCUM_RSRC_CMT_V	Returns current project– and task–level resource commitment summary amounts by the following periods: inception–to–date, year–to–date, prior period, and period–to–date

Table 6 – 18 Status API Views (Page 1 of 3)

View	Description
PA_ACCUM_RSRC_COST_BGT_V	Returns project– and task–level resource cost budget summary amounts for the following periods: inception–to–date, year–to–date, prior period, period–to–date, and total
PA_ACCUM_RSRC_REV_BGT_V	Returns project– and task–level resource revenue budget summary amounts for the following periods: inception–to–date, year–to–date, prior period, period–to–date, and total
PA_ACCUM_WBS_ACT_V	Returns current project– and task–level actual cost and revenue summary amounts by the following periods: inception–to–date, year–to–date, prior period, and period–to–date
PA_ACCUM_WBS_CMT_V	Returns current project– and task–level commitment summary amounts by the following periods: inception–to–date, year–to–date, prior period, and period–to–date
PA_ACCUM_WBS_COST_BGT_V	Returns project– and task–level cost budget summary amounts for the following periods: inception–to–date, year–to–date, prior period, period–to–date, and total
PA_ACCUM_WBS_REV_BGT_V	Returns project– and task–level revenue budget summary amounts for the following periods: inception–to–date, year–to–date, prior period, period–to–date, and total
PA_ACT_BY_GL_PERIOD_V	Returns actual cost and revenue totals for lowest tasks and resources by GL periods
PA_ACT_BY_PA_PERIOD_V	Returns actual cost and revenue totals for lowest tasks and resources by PA periods
PA_BURDEN_COMPONENT_CMT_V	Returns commitment burden components by resource, PA period name, expenditure type, expenditure organization, and burden set for each transaction summarization record
PA_BURDEN_COMPONENT_COST_V	Returns actual burden components by resource, PA period name, expenditure type, expenditure organization, and burden set for each transaction summarization record. This view returns burden cost components only for resources that have been burdened.
PA_CMT_BY_GL_PERIOD_V	Returns current commitment totals for lowest tasks and resources by GL period.
PA_CMT_BY_PA_PERIOD_V	Returns current commitment totals for lowest tasks and resources by PA periods
PA_GL_PERIODS_V	A view of the PA_PERIODS tables for GL periods and their start and end dates
PA_PA_PERIODS_V	A view of the PA_PERIODS tables for PA periods and their start and end dates

Table 6 – 18 Status API Views (Page 2 of 3)

View	Description
PA_PM_REFERENCE_V	Retrieves Oracle Projects identifiers and reference codes from your external systems for projects and tasks
PA_TXN_ACCUM_V	Shows detail information by various transaction attributes. Transaction attributes can include person, job, organization, vendor, expenditure type, event type, non–labor resource, expenditure category, revenue category, non–labor resource organization, event type classification, system linkage function, and week ending date.

Table 6 – 18 Status API Views (Page 3 of 3)

Status API Procedure Definitions

This section contains description of the status APIs, including business rules and parameters.

UPDATE_EARNED_VALUE

UPDATE_EARNED_VALUE is a PL/SQL procedure that updates earned value information in the PA_EARNED_VALUES table for lowest task–resource combinations. You can also use this procedure to update project–task rows.

Business Rules

- This procedure creates a new row in the table PA_EARNED_VALUES. CURRENT_FLAG is always set to Y for the last row inserted for each project, task, and resource combination. CURRENT_FLAG for all other rows is set to N.
- To create a project–task row, pass zero for the RESOURCE_LIST_MEMBER_ID parameter. To create a project row, pass zero for both the TASK_ID and RESOURCE_LIST_MEMBER_ID parameters.

  Note: This API assumes that the vendor of the external system maintains the appropriate earned value data for all levels in any given project.

The following table shows the parameters for UPDATE_EARNED_VALUE.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API version number
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Initial message table (default = 'F')
P_COMMIT	IN	VARCHAR2(1)	No	Commit (default = 'F')
P_RETURN_STATUS	OUT	VARCHAR2(1)		Return status
P_MSG_COUNT	OUT	NUMBER		Message count
P_MSG_DATA	OUT	VARCHAR2(2000)		Message
P_PROJECT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that identifies the project in the external system

Name	Usage	Type	Req?	Description
P_TASK_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
P_RESOURCE_LIST_MEMBER_ID	IN	NUMBER	Yes	The identification code of the resource list member
P_RESOURCE_ALIAS	IN	VARCHAR2(30)	No	The alias of the resource
P_RESOURCE_LIST_NAME	IN	VARCHAR2(60)	No	The name of the resource list
P_AS_OF_DATE	IN	DATE	Yes	As-of date
P_BCWS_CURRENT	IN	NUMBER	No	Budget cost of work performed
P_ACWP_CURRENT	IN	NUMBER	No	Actual cost of work performed
P_BCWP_CURRENT	IN	NUMBER	No	Budget cost of work performed
P_BAC_CURRENT	IN	NUMBER	No	Budget cost at completion
P_BCWS_ITD	IN	NUMBER	Yes	Inception-to-date budget cost of work performed
P_ACWP_ITD	IN	NUMBER	Yes	Inception-to-date actual cost of work performed
P_BCWP_ITD	IN	NUMBER	Yes	Inception-to-date budget cost of work performed
P_BAC_ITD ITD	IN	NUMBER	Yes	Inception-to-date budget cost at completion
P_BQWS_CURRENT	IN	NUMBER	No	Budget quantity of work performed
P_AQWP_CURRENT	IN	NUMBER	No	Actual quantity of work performed
P_BQWP_CURRENT	IN	NUMBER	No	Budget quantity of work performed
P_BAQ_CURRENT	IN	NUMBER	No	Budget quantity at completion
P_BQWS_ITD	IN	NUMBER	Yes	Inception-to-date budget quantity of work performed
P_AQWP_ITD	IN	NUMBER	Yes	Inception-to-date actual quantity of work performed
P_BQWP_ITD	IN	NUMBER	Yes	Inception-to-date budget quantity of work performed
P_BAQ_ITD	IN	NUMBER	Yes	Inception-to-date budget quantity at completion

UPDATE_PROGRESS

UPDATE_PROGRESS is a PL/SQL procedure that updates progress information in the PA_PERCENT_COMPLETES table as of a given date for all levels of the work breakdown structure.

For a given project, a task identifier of zero shows that the parameters apply to a project–level row. A task identifier greater than zero shows that the parameters apply to a task–level row.

Adding or deleting tasks from a project's work breakdown structure does not affect their corresponding rows in the PA_PERCENT_COMPLETES table. When executed, this API inserts a new row in the PA_PERCENT_COMPLETES table if a row for that project–task combination does not already exist.

Business Rules

- This procedure creates a new row in the table PA_PERCENT_COMPLETES. CURRENT_FLAG is always set to Y for the last row inserted for each project, task, and resource combination. CURRENT_FLAG for all other rows is set to N.
- To create a project row, you must pass zero in TASK_ID.

  Note: This API assumes that vendor of the external system maintains the appropriate rollup of progress data for each level of the work breakdown structure for any given project. Providing progress information, however, is optional.

The following table shows the parameters for UPDATE_PROGRESS.

Name	Usage	Type	Req?	Description
P_API_VERSION_NUMBER	IN	NUMBER	Yes	API version number
P_INIT_MSG_LIST	IN	VARCHAR2(1)	No	Initial message table (default = 'F')
P_COMMIT	IN	VARCHAR2(1)	No	Commit (default = 'F')
P_RETURN_STATUS	OUT	VARCHAR2(1)		Return status
P_MSG_COUNT	OUT	NUMBER		Message count
P_MSG_DATA	OUT	VARCHAR2(2000)		Message
P_PROJECT_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the project in Oracle Projects
P_PM_PROJECT_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the project in the external system
P_TASK_ID	IN	NUMBER	Yes	The reference code that uniquely identifies the task within a project in Oracle Projects
P_PM_TASK_REFERENCE	IN	VARCHAR2(30)	No	The reference code that uniquely identifies the task in the external system
P_AS_OF_DATE	IN	DATE	Yes	As-of-date
P_PERCENT_COMPLETE	IN	NUMBER	Yes	Percent complete

Custom Summarization Reporting APIs

The Custom Summarization Reporting APIs give you added control for custom summarization reporting.

Actuals Summarization API

You can use the Actuals Summarization API to get amounts by a specific Oracle Projects or Oracle General Ledger period, a specific range of Oracle Projects or Oracle General Ledger periods and by various transaction attributes as follows:

- Project, task, and resource combinations
- All levels of the project work breakdown structure
- Oracle Projects or Oracle General Ledger period
- Oracle Projects or Oracle General Ledger period ranges
- Various transaction attributes from the following:
  - employee
  - job
  - organization
  - supplier
  - expenditure type
  - event type
  - non–labor resource
  - expenditure category
  - revenue category
  - non–labor resource organization
  - event type classification
  - expenditure type classification

The name of the package is pa_accum_api, and the name of the procedure is get_proj_accum_actuals. You can get actual amounts only if you successfully ran the Update Project Summary process for the project.

This procedure returns the actual cost, revenue, and commitment amounts by:

Package.Procedure

The following table lists the parameters that Oracle Projects provides for the API pa_accum_api.get_project_accum_actuals. (See files PAAAPIS.pls and PAAAPIB.pls under the admin directory.)

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	The identifier for the project
X_task_id	IN	NUMBER	The identifier for the task. When retrieving project–level amounts, this parameter can be null.
X_resource_list_member_id	IN	NUMBER	The identifier for the resource. When retrieving aggregate project or task amounts, this parameter can be null.
X_period_type	IN	VARCHAR2	The identifier for the period type. You use this parameter to tell the procedure that you expect summary amounts by either PA or GL period. Allowable values are as P for PA Period, G for GL Period
X_from_period_name	IN	VARCHAR2	The identifier for the start period. You must pass this parameter and the parameter X_to_period_Name to the procedure with the following constraints: If the period type is PA, then both period names must be Oracle Projects period names. Otherwise, both period names must be GL period names. If you are retrieving amounts for one period, you must specify the same period name for both parameters. For a range of periods, the from period name must be earlier than the to period name
X_to_period_name	IN	VARCHAR2	The identifier for the end period. (See X_from_period_name for constraints about this parameter.)
X_person_id	IN	VARCHAR2	The identifier for the employee transaction attribute
X_job_id	IN	NUMBER	The identifier for the job transaction attribute
X_organization_id	IN	NUMBER	The identifier for the organization transaction attribute
X_vendor_id	IN	VARCHAR2	The identifier for the supplier transaction attribute
X_expenditure_type	IN	VARCHAR2	The identifier for the expenditure type transaction
X_event_type	IN	VARCHAR2	The identifier for the event type transaction attribute
X_non_labor_resource	IN	VARCHAR2	The identifier for the non–labor resource transaction attribute
X_expenditure_category	IN	VARCHAR2	The identifier for the expenditure category transaction attribute
X_revenue_category	IN	VARCHAR2	The identifier for the revenue category transaction attribute
X_non_labor_resource_org_id	IN	NUMBER	The identifier for the non–labor resource organization transaction attribute
X_event_type_classification	IN	VARCHAR2	The identifier for the event type classification transaction attribute

Table 6 – 19 (Page 1 of 2)

Parameter	Usage	Type	Description
X_system_linkage_function	IN	VARCHAR2	The identifier for the expenditure type class function transaction attribute
X_week_ending_date	IN	DATE	The identifier for the week ending date transaction attribute
X_revenue	INOUT	NUMBER	revenue amount
X_raw_cost	INOUT	NUMBER	raw cost amount
X_burdened_cost	INOUT	NUMBER	burdened cost amount
X_quantity	INOUT	NUMBER	quantity
X_labor_hours	INOUT	NUMBER	labor hours
X_billable_raw_cost	INOUT	NUMBER	billable raw cost amount
X_billable_burdened_cost	INOUT	NUMBER	billable burdened cost amount
X_billable_quantity	INOUT	NUMBER	billable quantity
X_billable_labor_hours	INOUT	NUMBER	billable labor hours
X_cmt_raw_cost	INOUT	NUMBER	commitment raw cost amount
X_cmt_burdened_cost	INOUT	NUMBER	commitment burdened cost amount
X_unit_of_measure	INOUT	VARCHAR2	unit of measure
X_err_stage	INOUT	VARCHAR2	error stage
X_err_code	INOUT	NUMBER	error code

Table 6 – 19 (Page 2 of 2)

Custom Reporting Strategies for the Actuals Summarization API

To see how the API can be used for reporting, refer to the following Oracle Projects reports:

- Revenue, Cost, Budgets by Resources (Project Level)
- Task – Revenue, Cost, Budgets by Resources

Budget Summarization API

You can use the Budget Summarization API for custom reporting. This API gets budget data for any baselined budget. You can get the budget data without running the Update Project Summary process.

The Budget API returns budget amounts by:

- Project, task, and resource combinations
- All levels of the project work breakdown structure
- All levels of the resource breakdown structure
- Oracle Projects or Oracle General Ledger period
- Oracle Projects or Oracle General Ledger period ranges
- Budget type

The Budget API can return summary amounts for budgets assigned to any level of the project and task work breakdown structure, providing you pass the task_id corresponding to the budgeted level to the Budget API. For example, if a project is budgeted at the top task and you pass a lower task to the Budget API, the Budget API will return zero budget amounts.

The name of the summarization package is **pa_accum_api** and the name of the budget procedure is **get_proj_accum_budgets**.

Package.Procedure

The following table lists the parameters that Oracle Projects provides for the Budget API, **pa_accum_api.get_proj_accum_budgets**.

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	The identifier of the project.
X_task_id	IN	NUMBER	The identifier of the task. Set to zero if budgeting at the project level.
X_resource_list_member_id	IN	NUMBER	The identifier of the resource list member. The value is null for project and task combinations.
X_period_type	IN	VARCHAR2	The identifier for the PA or GL Period: 'P' for PA periods or 'G' for GL periods.
X_from_period_name	IN	VARCHAR2	The start period of the period range
X_to_period_name	IN	VARCHAR2	The end period of the period range.

Table 6 – 20 (Page 1 of 2) Budget API Parameters

Parameter	Usage	Type	Description
X_budget_type_code	IN	VARCHAR2	The identifier of the budget type associated with the budget columns.
X_base_raw_cost	INOUT	NUMBER	Baseline raw cost budget
X_base_burdened_cost	INOUT	NUMBER	Baseline burdened cost budget
X_base_revenue	INOUT	NUMBER	Baseline revenue budget
X_base_quantity	INOUT	NUMBER	Baseline quantity budget. This column returns zero for project and task level combinations.
X_base_labor_quantity	INOUT	NUMBER	Baseline labor quantity
X_unit_of_measure	INOUT	VARCHAR2	Unit of measure. If the API finds multiple values, this column returns null. Otherwise, this column returns the unit of measure.
X_orig_raw_cost	INOUT	NUMBER	Original raw cost budget
X_orig_burdened_cost	INOUT	NUMBER	Original burdened cost budget
X_orig_revenue	INOUT	NUMBER	Original revenue budget
X_orig_quantity	INOUT	NUMBER	Original quantity budget
X_orig_labor_quantity	INOUT	NUMBER	Original labor quantity
X_err_stage	INOUT	VARCHAR2	Error stage
X_err_code	INOUT	NUMBER	Error code

Table 6 – 20 (Page 2 of 2) Budget API Parameters

PART III: CLIENT EXTENSIONS

CHAPTER

7

Overview of Client Extensions

This chapter describes everything you need to know about designing and writing client extensions in Oracle Projects.

Client Extensions

Use client extensions to extend the functionality of Oracle Projects. You can automate your company's business rules within the standard processing flow of Oracle Projects, without having to customize the software.

Alphabetical List of Client Extensions

The following table lists the client extensions. The package specification and body (template procedure) files are stored in the Oracle Projects patch / 115 / sql directory.

Client Extension	Package Specification File	Package Body File
Allocation: page 9 – 41	PAPALCCS.pls	PAPALCCB.pls
AR Transaction Type: page 10 – 68	PAXITRXS.pls	PAXITRXB.pls
Archive Custom Tables: page 8 – 34	PAXAPVXS.pls	PAXAPVXB.pls
Archive Project Validation: page 8 – 32	PAXAPVXS.pls	PAXAPVXB.pls
Asset Allocation Basis: page 9 – 54	PACCXAAS.pls	PACCXAAB.pls
Asset Assignment: page 9 – 57	PAPGALCS.pls	PAPGALCB.pls
Asset Lines Processing: page 9 – 60	PACCXACS.pls	PACCXACB.pls
Assignment Approval Changes: page 11 – 2	PARAAPCS.pls	PARAAPCB.pls
Assignment Approval Notification: page 11 – 4	PARAWFCS.pls	PARAWFCB.pls
AutoApproval: page 9 – 17	PAXPTEES.pls	PAXPTEEB.pls
Automatic Invoice Approve / Release: page 10 – 68	PAXPIACS.pls	PAXPIACB.pls
Billing Cycle: page 10 – 5	PAXIBCXS.pls	PAXIBCXB.pls
Billing Extensions: page 10 – 7	PAXITMPS.pls	PAXITMPB.pls
Budget Calculation: page 12 – 5	PAXBCECS.pls	PAXBCECB.pls
Budget Verification: page 12 – 13	PAXBCECS.pls	PAXBCECB.pls

Table 7 – 1 Client Extensions (Page 1 of 3)

Client Extension	Package Specification File	Package Body File
Budget Workflow: page 12 – 16	PAWFBCES.pls	PAWFBCEB.pls
Burden Costing: page 9 – 39	PAXCCEBS.pls	PAXCCEBB.pls
Capital Event Processing: page 9 – 62	PACCXCBS.pls	PACCXCBB.pls
Capitalized Interest: page 9 – 64	PACINTXS.pls	PACINTXB.pls
CIP Account Override: page 9 – 76	PACCXCOS.pls	PACCXCOB.pls
CIP Grouping: page 9 – 72	PAXGCES.pls	PAXGCEB.pls
Commitment Changes: page 8 – 19	PACECMTS.pls	PACECMTB.pls
Control Item Document Numbering: page 12 – 21	PACINRXS.pls	PACINRXB.pls
Cost Accrual Billing: page 10 – 48	PAXICOSS.pls	PAXICOSB.pls
Cost Accrual Identification: page 10 – 52	PAICPCAS.pls	PAICPCAB.pls
Cross Charge Client Extensions:		
Provider and Receiver Organizations Override: page 9 – 81	PACCIXTS.pls	PACCIXTB.pls
Cross–Charge Processing Method Override: page 9 – 83	PACCIXTS.pls	PACCIXTB.pls
Transfer Price Determination: page 9 – 86	PAPTPRCS.pls	PAPTPRCB.pls
Transfer Price Override: page 9 – 89	PAPTPRCS.pls	PAPTPRCB.pls
Transfer Price Currency Conversion Override: page 9 – 92	PAPMCECS.pls	PAPMCECB.pls
Cost Accrual Identification: page 10 – 52	PAICPCAS.pls	PAICPCAB.pls
Depreciation Account Override: page 9 – 78	PAXCXDES.pls	PAXCXDEB.pls
Descriptive Flexfield Mapping: page 8 – 28	PAPDFFCS.pls	PAPDFFCB.pls
Funding Revaluation Factor:: page 10 – 2	PAXBFRCS.pls	PAXBFRCB.pls
Issue and Change Workflow: page 12 – 23	PACIWFCS.pls	PACIWFCB.pls
Labor Billing: page 10 – 53	PAXICTMS.pls	PAXICTMB.pls
Labor Costing: page 9 – 19	PAXCCECS.pls	PAXCCECB.pls

Table 7 – 1 Client Extensions (Page 2 of 3)

Client Extension	Package Specification File	Package Body File
Labor Transaction: page 9 – 22	PAXCCETS.pls	PAXCCETB.pls
Overtime Calculation: page 9 – 33	PAXDLCOS.pls	PAXDLCOB.pls
Output Tax: page 10 – 64	PAXPOTXS.pls	PAXPOTXB.pls
Post–Import: page 8 – 26	PAXTTRXS.pls	PAXTTRXB.pls
Pre–Import: page 8 – 23	PAXTTRXS.pls	PAXTTRXB.pls
Project and Task Date: page 8 – 8	PAPMGCES.pls	PAPMGCEB.pls
Project Security: page 8 – 2	PAPSECXS.pls	PAPSECXB.pls
Project Status Inquiry (PSI): page 12 – 29	PAXVPS2S.pls	PAXVPS2B.pls
Project Status Report Workflow: page 12 – 26	PAPRWFCS.pls	PAPRWFCB.pls
Project Verification: page 8 – 5	PAXPCECS.pls	PAXPCECB.pls
Project Workflow: page 8 – 12	PAWFPCES.pls	PAWFPCEB.pls
Receivables Installation Override: page 10 – 66	PAPARICS.pls	PAPARICB.pls
Transaction Control: page 9 – 2	PAXTTCXS.pls	PAXTTCXB.pls
Verify Organization Change: page 8 – 15	PAXORCES.pls	PAXORCEB.pls
Workplan Workflow: page 12 – 2	PAXSTWCS.pls	PAXSTWCB.pls

Table 7 – 1 Client Extensions (Page 3 of 3)

You use PL/SQL to modify procedures within the extensions. Oracle Projects calls these procedures during specific points in the standard processing.

The procedures that you write are *extensions*, not *customizations*. Extensions are supported features within the product and are easily upgraded between product releases. Customizations are changes to the base product which are not supported and are not easily upgraded.

Warning: Do not insert or update records directly into any Oracle Applications table; using extensions to do so is not supported by Oracle Corporation. You must use the public, predefined procedures that Oracle Projects provides to insert or update records in Oracle Projects tables. You are responsible for the support and upgrade of the procedures that you write that are affected by changes between releases of Oracle Applications.

Implementing Client Extensions

To implement client extensions, you must analyze your business requirements, design the client extension logic, and then write the appropriate PL/SQL procedures. Each of these steps is described in this section.

Each step requires a specific expertise. The analysis and design portions require an implementation team member who knows company's business rules, how Oracle Projects is set up in your company, and how you want to use the client extensions. The PL/SQL coding portion requires a team member who is adept with PL/SQL and the Oracle Projects data structures. Typically, the implementation team includes two or more people working together to provide the necessary expertise.

Analyzing Your Business Requirements

First determine if you need to use client extensions at all.

1. Define and document your company's business requirements and rules.
2. Determine if these business rules are handled by the standard features of Oracle Projects.
3. For those business rules not handled by the standard functionality, determine which client extensions can address your specific business needs.

Example

Your company has defined a policy that supplies must be charged to overhead projects.

You review your implementation of Oracle Projects and find that you can use transaction controls to specify what can be charged to a specific project or task. The rule regarding supplies is applicable to all projects that are not overhead projects. You decide it is impractical to implement this rule by defining transaction controls for every non–overhead project.

You decide to use transaction control extensions to implement this policy.

Designing the Logic

Careful design is critical. If you create careful, thorough design and specifications in this stage, you can expect more ease in writing the PL/SQL procedure and a more successful client extension implementation. This design cycle includes the following steps:

1. Understand the client extensions you propose to use, including their purpose, processing flow, when Oracle Projects calls the extensions, and the input values. For a list and reference for each extension, see: Alphabetical List of Client Extensions: page 7 – 2.
2. Define and document the requirements and logic of your business rules under all possible conditions. Determine the inputs, calculations performed, and resulting outputs.
3. Determine the data elements required to enforce your rules and how you will select or derive each of the required elements. Define additional implementation data and document additional business procedures based on the requirements of your business rules.
4. Step through various business scenarios to ensure that your logic handles each condition as you expect. You can use these scenarios as test cases when you test your actual client extension definition and procedure.
5. Give the detailed specification to the team member who will write the PL/SQL procedure.

   Note: If you want to use different logic for different parts of your enterprise, write one procedure that branches appropriately.

Determining Data Elements

Each client extension contains predefined parameters. The program that calls and executes the client extension passes in values for the predefined parameters.

You can derive additional parameters from the predefined parameters. For example, if a client extension has a predefined parameter of PROJECT_ID (project identifier), you can derive the project type from PROJECT_ID.

You can also use descriptive flexfield segments to hold additional data as inputs to your rules. When you write the PL/SQL procedure, you select from the descriptive flexfield segment column that holds the appropriate input value.

You can derive data for any Oracle table as input into your rules, as long as you can derive the values from the predefined input values passed into the PL/SQL procedure.

Example: Designing a Client Extension

Let's use our earlier transaction control extension example to illustrate these design steps. (See: Analyzing Your Business Requirements: page 7 – 5.)

1. After studying transaction control extensions, you decide to use the transaction control extensions so that users can charge supplies only to overhead projects.
2. You define the logic for the transaction control extension as:

```
IF     charging supplies
THEN   IF     charging to overhead projects
       THEN   OK
       ELSE   error message
You can charge supplies only to overhead projects
ELSE   OK
```

3. You determine the data elements that identify which transactions are supplies and which projects are overhead projects.

   You decide that the expenditure type of *Supplies* specifies the type of charge, and that the project type of *Overhead* specifies the type of project.

   The predefined parameters for the extension include expenditure type (*Supplies*) and project ID. You can derive the project type (*Overhead*) from the project ID.

   The logic is:

```
IF Expenditure Type = Supplies
THEN   IF Project Type = Overhead
       THEN   OK
       ELSE   error message
You can charge supplies only to overhead projects
ELSE   OK
```

4. You step through several scenarios using different types of charges and different types of projects. Your logic handles all of the scenarios.
5. You are ready to hand off this specification to your technical resource.

Writing PL/SQL Procedures

This section is a brief overview of PL/SQL procedures. For more information, see: *PL/SQL User's Guide and Reference Manual*.

> **Note:** We recommend that you keep the *PL/SQL User's Guide and Reference Manual* on hand as reference material while defining procedures. In addition, you can refer to the Oracle eTechnical Reference Manuals (eTRM) for detailed description of database tables and views. Oracle eTRM is available on *OracleMetaLink*.

Packages

Packages are database objects that group logically related PL/SQL types, objects, and subprograms. Packages usually consist of two files: a package specification file and a package body file. The files are described in the following table:

File	Description
Package Specification File	The specification file is the interface to your applications. It declares the types, variables, constants, exceptions, cursors, and subprograms available for use in the package. In Oracle Projects client extensions, this file contains the package name, procedures, and function declarations. If you create procedures within the package outside the predefined procedure, you must also modify this file.
Package Body File	The package body contains the actual PL/SQL code used to implement the business logic. In Oracle Projects client extensions, this file contains the procedure or procedures that you modify to implement the extension. You can define as many procedures as you like within the package or within the predefined procedure or procedures.

Table 7 – 2 (Page 1 of 1)

Warning: Do not change the name of the extension procedures. In addition, do not change the parameter names, parameter types, or parameter order in your procedure.

Suggestion: After you write a procedure, do not forget to compile it and store it in the database.

Procedures

Procedures are subprograms within a package. Procedures are invoked by the application and perform a specific action. Procedures define what parameters will be passed in as context for the program, how the inputs are processed, and what output is returned. A procedure consists of the following elements:

Inputs Each procedure has predefined input parameters, which must be passed in the predefined order. The parameters identify the transaction being processed and the context in which the program is called. You can derive additional inputs from any Oracle table based on the predefined input parameters.

Logic The procedure uses the inputs and performs any logical processing and calculations. The program can be a simple program, such that it returns a fixed number, or it can be a complex algorithm that performs multiple functions.

Outputs Each procedure returns whatever value you define it to return. For example, your procedure for transaction control extensions could return a null value if the transaction passes all validation rules, or an error message if validation fails.

Syntax

A procedure consists of two parts: the *specification* and the *body*.

The procedure specification begins with the keyword PROCEDURE and ends with the procedure name or a parameter list.

The procedure body begins with the keyword IS and ends with the keyword END, followed by an optional procedure name. The procedure body has a declarative part, an executable part, and an optional error handling part.

You write procedures using the following syntax:

```
PROCEDURE name [  (parameter [, parameter,...])  ] IS
  [local declarations]
BEGIN
  executable statements
[EXCEPTION
```

```
    exception handlers]
END [name];
```

The parameter syntax above expands to the following syntax:

```
var_name [IN | OUT | IN OUT] datatype [{:= | DEFAULT} value]
```

For more information, refer to the *PL/SQL User's Guide and Reference Manual*.

Using Template Procedures

Oracle Projects provides you with template procedures for each client extension that you can use to write your own procedures. Each template procedure contains predefined parameters that are passed into the procedure by the program that calls the procedure; you cannot change these predefined input parameters.

The template procedure files are stored in the Oracle Projects patch/115/sql directory.

Review the appropriate files before you design and implement a client extension. They provide a lot of useful information, including the predefined input parameter list and example case studies.

Make copies of these template files in a directory used by your company to store code that you have written, and then modify the copies. These template files will be replaced when the software is upgraded between releases. Use your modified files to reinstall your procedures after an upgrade to a new release of Oracle Projects.

Writing Logic in Your PL/SQL Procedures

You write the logic in the PL/SQL procedures based on the functional specifications created during the design process. Before you write the client extension PL/SQL procedures, you should have a clear understanding of the client extension procedures; including the inputs and outputs, error handling, and any example procedures provided for each extension. Read the appropriate client extension essays and template procedures to obtain detailed information.

> **Note:** Do not commit data within your PL/SQL procedure. Oracle Projects processes that call your procedures handle the commit logic.

Compiling and Storing Your Procedures

After you write your procedures and ensure that the specification file correctly includes any procedures that you have defined, compile and store the procedures in the database in the Applications Oracle user name. Install the package specification first, and then install the package body.

The template procedure files include syntax for compiling and storing the PL/SQL procedures. Assuming you have written your procedures using copies of the template procedure files, change to the directory in which your files are stored (use the command that is appropriate to your operating system):

```
$ sqlplus <apps user name>/<apps password>
SQL> @<spec_filename>.pls
SQL> @<body_filename>.pls
```

For example, if your Oracle Applications Oracle user name/password is apps/apps, you could use the following commands to install your transaction control extensions:

```
$ sqlplus apps/apps
SQL> @PAXTTXCS.pls
SQL> @PAXTTXCB.pls
```

If you encounter errors when you are creating your packages and its procedures, correct the errors and recreate your packages. You must successfully compile and store your package and its procedures in the database before you can use the client extensions in Oracle Projects.

Testing Your Procedures

After you have created your client extension procedures, test your client extension definitions within the processing flow of Oracle Projects to verify that you get the expected results.

CHAPTER

8

Oracle Project Foundation Client Extensions

This chapter describes the client extensions in the Oracle Project Foundation application.

Project Security Extension

Oracle Projects provides a client extension, PA_SECURITY_EXTN, that enables you to override the default project–based security and implement your own business rules for project and labor cost security. For more information on project–based security, see Security in Oracle Projects, *Oracle Projects Fundamentals*. This extension applies only to Oracle Projects windows and not to reports. Some examples of rules that you may define are:

- Only users who belong to the same organization as the project organization can access the project (organization–based security). Sample code for this example is included in the client extension package.
- All project administrators can view and update projects to which they are assigned, but project managers can only view those projects to which they are assigned
- Some responsibilities can view or update only capital projects (for an environment where users who handle capital projects do not handle contract and indirect projects)

Considerations for Project Security Extension Logic

You should determine the logic and the additional data elements your client extension requires before you write it. We recommend that you consider the following design issues for the project security extension:

- What are the conditions or circumstances in which project or labor security is based? What types of users? How will you identify the users? What types of projects? How will you identify the projects?
- Do you want the users to view the project but not update it, or do you want to block the project from their online queries?
- Does the type of security for a given user or set of projects change depending on the module?
- How does project security interact with the function security defined for the responsibility?
- Consider the performance implications of the logic that you write. The extension is called for every project during online queries.

Writing the Project Security Extension

The extension is identified by the following items:

Item	Name
Body template	PAPSECXB.pls
Specification template	PAPSECXS.pls
Package	pa_security_extn
Procedure	check_project_access

Table 8 – 1

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Package.Procedure

The following table lists the parameters that Oracle Projects provides for the procedure **pa_security_extn.check_project_access**.

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	Identifier of the project or project template
X_person_id	IN	NUMBER	Identifier of the person
X_cross_project_user	IN	VARCHAR2	Indicates if the user has cross–project update access: Y / N
X_cross_project_view	IN	VARCHAR2	Indicates if the user has cross–project view access: Y / N
X_calling_module	IN	VARCHAR2	Module in which the project security extension is called; Oracle Projects sets this value for each module in which it calls the security extension. The values are listed below.

Table 8 – 2 (Page 1 of 2) Project Security Extension Parameters

Parameter	Usage	Type	Description
X_event	IN	VARCHAR2	Type of query level to check upon which you can define specific rules: ALLOW_QUERY ALLOW_UPDATE VIEW_LABOR_ COSTS
X_value	OUT	VARCHAR2	Values to specify if result of the event: Y / N

Table 8 – 2 (Page 2 of 2) Project Security Extension Parameters

Additional Information about Parameters

The parameter X_calling_module allows you to write security rules based on the module in which the extension is called. The values are as follows:

Value	Description
PAXBUEBU	Budgets window
PAXCARVW	Capital Projects window
PAXINEAG	Agreements window
PAXINRVW	Invoice Review window
PAXINVPF	Project Funding Inquiry window
PAXPREPR	Projects window
PAXTRAPE.PROJECT	Project Expenditure Inquiry window
PAXURVPS	Project Status Inquiry window

Table 8 – 3 (Page 1 of 1) Additional Information About Parameters

Refer to the PA_Security_Extn procedure for the most up–to–date information about values for X_calling_module.

Project Verification Extension

The Project verification extension contains procedures that enable you to define rules for the following purposes:

- To determine whether a project can change its project status
- To determine whether to call Workflow for a project status change

Processing

Oracle Projects calls the Project Verification Extension when a change of status is requested for a project.

Designing Project Verification Extensions

You must determine what business rules you want to apply when a project status change is selected for a project. See also: Project Statuses, *Oracle Projects Implementation Guide*.

Writing Project Verification Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXPCECB.pls
Specification template	PAXPCECS.pls
Package	pa_client_extn_proj_status

Table 8 – 4

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

verify_project_status_change

Use this procedure to define requirements a project must satisfy to change from one project status to another. Detailed instructions for modifying the procedure are included in the package body.

The following table lists the parameters that Oracle Projects provides for the verify_project_status_change procedure.

Parameter	Usage	Type	Description
x_calling_module	IN	VARCHAR2	The module that called the extension.
x_project_id	IN	NUMBER	Identifier of the project.
x_old_proj_status_code	IN	VARCHAR2	The current project status code.
x_new_proj_status_code	IN	VARCHAR2	The new project status code.
x_project_type	IN	VARCHAR2	The project type of the project.
x_project_start_date	IN	DATE	The project start date.
x_project_end_date	IN	DATE	The project end date.
x_public_sector_flag	IN	VARCHAR2	Public sector indicator.
x_attribute_category	IN	VARCHAR2	Descriptive flexfield context.
x_attribute1 through x_attribute10	IN	VARCHAR2	Descriptive flexfield segments.
x_pm_product_code	IN	VARCHAR2	The project management product code.

Table 8 – 5 (Page 1 of 2) Verify Project Status Change Parameters

Parameter	Usage	Type	Description
x_err_code<	OUT	NUMBER	Error handling code.
x_warnings_only_flag	OUT	VARCHAR2	

Table 8 – 5 (Page 2 of 2) Verify Project Status Change Parameters

check_wf_enabled

When Oracle Projects determines whether to call Workflow for a project status change, it bases the decision on the settings in the project status record and the project type. You can use this procedure to override those settings and / or add additional requirements.

The following table lists the parameters that Oracle Projects provides for the check_wf_enabled procedure.

Parameter	Usage	Type	Description
x_project_status_code	IN	VARCHAR2	The current project status code
x_project_type	IN	VARCHAR2	The project type of the project
x_project_id	IN	NUMBER	Identifier of the project
x_wf_enabled_flag	OUT	VARCHAR2	Flag indicating whether Workflow is enabled for the status change. Value is either *Y* or *N*.
x_err_code	OUT	NUMBER	Error handling code
x_status_type	IN	VARCHAR2	The project status type

Table 8 – 6 (Page 1 of 1) Check Workflow Enabled Parameters

Project and Task Date Client Extension

You can customize this client extension to substitute dates used by external systems for the standard Oracle Projects project and task start and completion dates.

Oracle Projects supports the following project tracking dates through the Oracle Projects APIs. When you download a project from an external system, you can pass the values for these dates and store them in Oracle Projects as the project and task start and completion dates.

- Actual start date
- Actual finish date
- Early start date
- Early finish date
- Late start date
- Late finish date
- Scheduled start date
- Scheduled finish date

The extension is identified by the following items:

Item	Name
Body template	PAPMGCEB.pls
Specification template	PAPMGCES.pls
Package	pa_client_extn_pm
Procedure	customize_dates

Table 8 – 7

The template package contains default logic to return the date information that was passed to the API without substituting it for the Oracle Projects project or task start or completion date.

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Writing the Project and Task Date Client Extension

The customize dates procedure is described below.

Package.Procedure: pa_client_extn_pm.customize_dates

The following table lists the parameters provided by Oracle Projects for the project date client extension.

Parameter	Usage	Type	Required?	Description
P_PM_PROJECT_REFERENCE	IN	VARCHAR2	No	The reference code that uniquely identifies the project in the external system
P_PM_TASK_REFERENCE	IN	VARCHAR2	No	The reference code that uniquely identifies the task in the external system
P_PROJECT_ID	IN	NUMBER	No	The reference code that uniquely identifies the project in Oracle Projects
P_TASK_ID	IN	NUMBER	No	The reference code that uniquely identifies the task in Oracle Projects
P_PM_PRODUCT_CODE	IN	VARCHAR2	Yes	The product code of the external system
P_IN_START_DATE	IN	DATE	Yes	The default start date
P_IN_COMPLETION_DATE	IN	DATE	Yes	The default completion date
P_ACTUAL_START_DATE	IN	DATE	No	The actual start date
P_ACTUAL_FINISH_DATE	IN	DATE	No	The actual finish date
P_EARLY_START_DATE	IN	DATE	No	The early start date
P_EARLY_FINISH_DATE	IN	DATE	No	The early finish date
P_LATE_START_DATE	IN	DATE	No	The late start date
P_LATE_FINISH_DATE	IN	DATE	No	The late finish date

Table 8 – 8 (Page 1 of 2) Project and Task Date Parameters

Parameter	Usage	Type	Required?	Description
P_SCHEDULED_START_DATE	IN	DATE	No	The scheduled start date
P_SCHEDULED_FINISH_DATE	IN	DATE	No	The scheduled finish date
P_OUT_START_DATE	OUT	DATE		
P_OUT_COMPLETION_DATE	OUT	DATE		
P_ERROR_CODE	OUT	NUMBER		
P_ERROR_MESSAGE	OUT	VARCHAR2		

Table 8 – 8 (Page 2 of 2) Project and Task Date Parameters

You can customize this client extension to substitute a different set of project and task start dates for the standard Oracle Projects project and task start and completion dates. For example, you can define your own rules to determine which project and task dates in the external system correspond to the project and task start and completion dates in Oracle Projects.

The following code shows how to map the actual start and actual finish dates in an external system to the project and task start and completion dates in Oracle Projects.

> **Note:** The parameters P_OUT_START_DATE and P_OUT_COMPLETION_DATE must return valid values. The public APIs read the values and will not execute properly if the date values are invalid.

```
 -- Initialize the out variables
  p_error_code := 0;
  p_error_stage := NULL;
  IF p_actual_start_date IS NOT NULL and
p_actual_finish_date
       IS NOT NULL THEN
        p_out_start_date := p_actual_start_date;
        p_out_finish_date := p_actual_finish_date;
 ELSE
        p_out_start_date := p_in_start_date;
        p_out_completion_date := p_in_completion_date;
 END IF;
  -- To specify conditions based on different external
products that you
```

```
  -- are importing from, use code that looks something like
this
IF p_pm_product_code = <Your product code> THEN
      IF p_actual_start_date IS NOT NULL and
p_actual_finish_date IS NOT NULL THEN
           p_out_start_date := p_actual_start_date;
           p_out_finish_date := p_actual_finish_date;
      ELSE
            p_out_start_date := p_in_start_date;
            p_out_completion_date := p_in_completion_date;
      END IF;
ELSIF  p_pm_product_code = <differenct product code>
       IF p_early_start_date IS NOT NULL and
p_early_finish_date IS NOT NULL THEN
            p_out_start_date := p_early_start_date;
            p_out_finish_date := p_early_finish_date;
       ELSE
            p_out_start_date := p_in_start_date;
            p_out_completion_date := p_in_completion_date;
       END IF;
ELSE
        p_out_start_date := p_in_start_date;
        p_out_completion_date := p_in_completion_date;
END IF;
-- If you want different mappings for projects and tasks
then base your logic on
-- p_pm_task_reference or p_task_id
     IF (p_pm_task_reference IS NOT NULL or p_task_id IS
NOT NULL)  THEN
     -- (this means this is for a task)
           -- place the logic for assigning one set of
dates here
    ELSE -- ( this means this is for a project)
         -- place the logic for assigning a different set
of dates
    END IF;
EXCEPTION
      WHEN OTHERS THEN
            p_error_code := -1;
          -- If ORACLE error then set p_error_code to
SQLCODE
          -- Handle your exception here
```

Project Workflow Extension

The project workflow extension enables you to customize the workflow processes for changing project statuses.

You must determine how you want to identify the approver for a project status change. See also: Project Statuses: page, *Oracle Projects Implementation Guide.*

Processing

The default project workflow process calls the project workflow extension to determine the project approver.

Writing Project Workflow Extensions

The extension is identified by the following items:

Item	Name
Body template	PAWFPCEB.pls
Specification template	PAWFPCES.pls
Package	pa_client_extn_project_wf

Table 8 – 9

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Package Procedures

Following are the procedures included in the project workflow extension.

select_project_approver

This procedure returns the project approver ID to the calling workflow process. You can modify the procedure to add rules to determine who

can approve a project. The default procedure returns the ID of the supervisor of the person who submitted the project status change.

The following table lists the parameters that Oracle Projects provides for the select_project_approver procedure.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project.
p_workflow_ started_by_id	IN	NUMBER	Identifier of the person who submitted the project status change.
p_project_approver_ id	OUT	NUMBER	Identifier of the project approver.

Table 8 – 10 (Page 1 of 1) Select Project Approver Parameters

start_project_wf

This procedure starts the workflow process for project status changes.

The following table lists the parameters that Oracle Projects provides for the start_project_wf procedure.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project.
p_item_type	IN	VARCHAR2	The workflow item type.
p_process	IN	VARCHAR2	Name of the workflow process.
p_out_item_key	OUT	VARCHAR2	The workflow item key.
p_err_stack	OUT	VARCHAR2	Error handling stack.
p_err_stage	OUT	VARCHAR2	Error handling stage.

Table 8 – 11 (Page 1 of 2) Start Project Workflow Parameters

Parameter	Usage	Type	Description
p_err_code	OUT	VARCHAR2	Error handling code.
p_status_type	IN	VARCHAR2	The project status type

Table 8 – 11 (Page 2 of 2) Start Project Workflow Parameters

Verify Organization Change Extension

The Verify Organization Change Extension enables you to build business rules to determine whether an organization change is allowed for a Project / Task Owning Organization, and to define the error messages that are used when the rules are violated.

Processing

Oracle Projects calls the Verify Organization Change Extension during the Mass Update Batches process, and in the Projects window when the project or task owning organization is changed.

See Also

Writing the Verify Organization Change Extension

The extension is identified by the following items:

Item	Name
Body template	PAXORCEB.pls
Specification template	PAXORCES.pls
Package	pa_org_client_extn
Procedure	verify_org_change

Table 8 – 12

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Package Procedure

The verify organization change extension procedure is described below.

pa_org_client_extn.verify_org_change

The following table lists the parameters that Oracle Projects provides for the verify organization change extension.

Parameter	Usage	Type	Description
X_insert_update_ mode	IN	VARCHAR2	Value = *INSERT* if the project / task record has not been saved in the database. Value = *UPDATE* if the record exists in the database.
X_calling_module	IN	VARCHAR2	*PAXPREPR* if this extension is called from the Projects window. *PAXBAUPD* if this extension is called from the Process Mass Update Batches process.
X_project_id	IN	NUMBER	Identifier of the project to be updated.
X_task_id	IN	NUMBER	Identifier of the task to be updated. The value is NULL when the extension is called to verify a project organization change.
X_old_value	IN	NUMBER	Identifier of the current organization of the project or task.
X_new_value	IN	NUMBER	Identifier of the new organization to be assigned to the project or task.
X_project_type	IN	VARCHAR2	Identifier of the project type of the project.
X_project_start_date	IN	DATE	Start date of the project.
X_project_end_date	IN	DATE	End date of the project.
X_public_sector_flag	IN	VARCHAR2	Public sector flag on the project.

Table 8 – 13 (Page 1 of 2) Verify Organization Change Parameters

Parameter	Usage	Type	Description
X_task_manager_ person_id	IN	NUMBER	Identifier of the manager of the task.
X_service_type	IN	VARCHAR2	Service type code of the task.
X_task_start_date	IN	DATE	Start date of the task.
X_task_end_date	IN	DATE	End date of the task.
X_entered_by_user_ id	IN	NUMBER	Identifier of the user who entered the project / task.
X_attribute_category	IN	VARCHAR2	Attribute category of the project or task.
X_attribute_1 through X_attribute_10	IN	VARCHAR2	Attribute values 1 through 10 of the project or task.
X_pm_product_code	IN	VARCHAR2	Project management product code specified for the project or task.
X_pm_project_ reference	IN	VARCHAR2	Project management product reference specified for the project.
X_pm_task_ reference	IN	VARCHAR2	Project management task reference specified for the task.
X_functional_ security_flag	IN	VARCHAR2	Value = *Y* if the user's responsibility has the function Project: Org Update: Override Standard Checks. Otherwise, value = *N*.
X_outcome	OUT	VARCHAR2	The message error code if a verification rule is violated or if there is an Oracle error.

Table 8 – 13 (Page 2 of 2) Verify Organization Change Parameters

See Also

Function Security in Oracle Projects, *Oracle Projects Implementation Guide*

Commitment Changes Extension

When you run the PRC: Update Project Summary Amounts process, Oracle Projects checks commitments for each project to see if changes have occurred. If any of these changes have occurred, the commitment summary amounts are deleted and recreated.

If you have modified the Oracle Projects commitments view, PA_COMMITMENT_TXNS_V, you must also modify the Commitment Changes client extension to test for changes in commitments.

Writing the Commitment Changes Extension

The extension is identified by the following items:

Item	Name
Body template	PACECMTB.pls
Specification template	PACECMTS.pls
Package	pa_client_extn_check_cmt
Procedure	commitments_changed

Table 8 – 14

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Package.Function

Following are the procedures included in this extension.

pa_client_extn_check_cmt.commitments_changed

The body template includes a sample procedure that contains the default coding for the COMMITMENTS_CHANGED function. By default, the procedure checks for the following changes in the system–defined commitments view:

- new commitments have been added

- a commitment has been fully or partially converted to cost (for example, a purchase order has been matched by a supplier invoice.)
- the status of a commitment has changed from Unapproved to Approved

If the commitments have changed, then the function returns a value of Y. Otherwise, it returns the value N. If Y is returned, then the summarization process rebuilds the commitment summarization amounts.

If you have modified the commitments view, you must modify this procedure so that it can determine whether the user–defined commitments have changed from the last summarization process.

The sample procedure includes the following assumptions:

- The user commitment view is PA_COMMITMENTS_OUTSIDE_SYSTEM
- The line type is *I*
- The transaction source is OUTSIDE_SYSTEM
- The column CMT_HEADER_ID stores the header ID from the user view
- The column CMT_LINE_NUMBER stores the line number from the user view
- The APPROVED_FLAG is checked for a change since the last summarization process

The sample procedure checks for the following conditions:

- commitments in PA_COMMITMENT_TXNS with a different status (the APPROVED_FLAG column) from the same commitment in the User view
- commitments in the user view that do not exist in PA_COMMITMENT_TXNS

You must determine which column or columns in your commitments view to check for a change in value, and identify the procedure to check for new commitments.

Transaction Import Client Extensions

Use the Transaction Import Client Extensions to add procedures that run before or after the Transaction Import Process. The Transaction Import Process loads data from other applications into Oracle Projects. You can use the Pre–Import and Post–Import client extensions.

- Use the *Pre–Import Client Extension* to load the Transaction Interface Table (PA_TRANSACTION_INTERFACE_ALL) or to perform pre–import data validation.
- Use the *Post–Import Client Extension* to record the expenditure and expenditure item IDs generated by the Transaction Import Process in the source system. You can also use it for other post–import processing.

The Pre–Import and Post–Import client extensions are called depending upon the Transaction Source that is used in the Transaction Import Process. When you run the Transaction Import Process, you must specify a Transaction Source that determines how the Transaction Import processes the transactions. The Pre–Import and Post–Import client extensions are specified when you set up the transaction source in the Transaction Sources window. The following attributes of the transaction source are used:

- The *Pre Processing Extension* is where you specify the Pre–Import client extension.
- The *Post Processing Extension* is where you specify the Post–Import client extension.

  Note: For both the Pre and Post Processing Extensions, you enter the full name of the client extension, including the package, in the format *package.procedure*.

Each transaction source that you set up can use the same Pre–Import and Post–Import client extensions, or each transaction source can have unique Pre–Import and Post–Import client extensions. For example, if you set up a transaction source for importing data from an external accounts payable system and a different transaction source for importing data from an external time management system, you can create different Pre–Import client extensions for each of the transaction sources or use the same Pre–Import client extension for both transaction sources.

The Oracle Internet Time transaction source that is included with Oracle Projects comes with both a predefined Pre–Import client extension and a Post–Import client extension. These two client extensions are described in the following sections:

Pre–Import Client Extension for Internet Time: page 8 – 23

Post–Import Client Extension for Internet Time: page 8 – 26

You may refer to the existing client extensions for Internet Time when you create additional Pre–Import and Post–Import client extensions.

See Also

Transaction Import Interface: page 13 – 26

Transaction Source Options, *Oracle Projects Implementation Guide*

Pre–Import Client Extension for Internet Time

Use the Pre–Import Client Extension for Internet Time to load approved self–service time cards into the Oracle Projects Transaction Interface Table (PA_TRANSACTION_INTERFACE_ALL). Once data is loaded in the transaction interface table, the Transaction Import Process will load the data into Oracle Projects.

This client extension allows you to automate the process of loading Oracle Internet Time data to the interface table as part of the import process.

Oracle Projects calls the Pre–Import Client Extension for Internet Time at the beginning of the Transaction Import Process when you use the Oracle Internet Time transaction source.

If you specify a batch name when you run the Transaction Import Process, the Pre–Import Client Extension for Internet Time loads all data from the Oracle Projects Expenditures table with a matching batch name and with a transfer status code of *Pending*. If no batch name is entered, then all records marked as *Pending* are selected for interface.

The Pre–Import Client Extension for Internet Time loads Internet Time data into the interface table without performing any validation. Therefore, only system errors are expected. If a system error does occur, no transfer will take place and all data will remain in the Expenditures table with a status of *Pending*.

If all items are successfully loaded into the transaction interface table, then the Transaction Status Code in the Interface table for all items in the expenditure is set to *Pending*.

> **Note:** If you add validation logic to a custom extension, and the transaction fails the validation, then the Transaction Status Code is set to ”Failed Pre” for all items in the expenditure. The failed items will have to be fixed in the external system.

After the last item for an expenditure is successfully loaded into the Transaction Interface Table, then the Transfer Status Code in the Expenditures table is set to *Transferred*.

Description

The extension is identified by the following items:

Item	Name
Body template	PAXTTRXB.pls
Specification template	PAXTTRXS.pls
Package	pa_trx_import
Procedure	pre_import

Table 8 – 15

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Pre–Import Procedure

The pre–import procedure uses the following parameters:

Parameter	Usage	Type	Description
p_transaction_source	IN	VARCHAR2	Classification of the transactions loaded into Oracle Projects from an external system.
p_batch	IN	VARCHAR2	User entered name for grouping expenditures within a transaction source.
p_xface_id	IN	NUMBER	System–generated number that identifies all the transactions processed by a given concurrent request.
p_user_id	IN	NUMBER	User.

Table 8 – 16 Pre–import parameters (Page 1 of 1)

See Also

Transaction Import Client Extensions: page 8 – 21

Transaction Import Interface: page 13 – 26

Transaction Sources, *Oracle Projects Implementation Guide*

Transaction Source Options, *Oracle Projects Implementation Guide*

Post–Import Client Extension for Internet Time

Use the Post–Import Client Extension for Internet Time to tie back the Oracle Internet Time records that have been imported into Oracle Projects to the source transactions in Oracle Internet Time.

Oracle Projects calls the Post–Import Client Extension for Internet Time after the Transaction Import Process runs when you use the Oracle Internet Time transaction source.

If all items within an expenditure pass through the Post–Import extension successfully, then the Transaction Status Code in the Interface table for all of the items in the expenditure is set to *Accepted*. If any one of the items in the expenditure fails, then the Transaction Status Code for all items in the expenditure is set to *Failed Post*. These records are processed again the next time the transaction import is run for the batch.

In Internet Time, only system errors are expected during the Post–Import processing. If a system error occurs, then the Transfer Status Code in the Expenditures table will remain *Transferred* and the Transaction Status Code in the Interface table remains *Imported*. They are processed again the next time you run the Post–Import process.

Description

The extension is identified by the following items:

Item	Name
Specification template	PAXTTRXS.pls
Body template	PAXTTRXB.pls
Package	pa_trx_import
Procedures	post_import

Table 8 – 17 Post–Import Extension

Post–Import Procedure

The post–import procedure uses the following parameters:

Parameter	Usage	Type	Description
p_transaction_source	IN	VARCHAR2	Classification of the transactions loaded into Oracle Projects from an external system.
p_batch	IN	VARCHAR2	User entered name for grouping expenditures within a transaction source.
p_xface_id	IN	NUMBER	System–generated number that identifies all the transactions processed by a given concurrent request.
p_user_id	IN	NUMBER	User.

Table 8 – 18 Post–import parameters (Page 1 of 1)

See Also

Transaction Import Client Extensions: page 8 – 21

Transaction Import Interface: page 13 – 26

Transaction Sources, *Oracle Projects Implementation Guide*

Transaction Source Options, *Oracle Projects Implementation Guide*

Descriptive Flexfield Mapping

Use the Descriptive Flexfield Mapping client extension to map segments of descriptive flexfield that are transferred from Oracle Payables to Oracle Projects or from Oracle Projects to Oracle Payables.

To transfer descriptive flexfields between Oracle Projects and Oracle Payables, you must set the *PA: Transfer DFF with AP* profile option to *Yes*. When this profile option is set, Oracle Projects calls the Descriptive Flexfield Mapping extension during the processes that interface transactions between the two applications.

You can modify the extension to customize how descriptive flexfields are mapped when they are transferred.

> **Note:** Projects holds 10 descriptive flexfield segments. If you are using more than 10 segments in Payables, only the first 10 are imported to Projects.

Description

The extension is identified by the following items:

Item	Name
Specification template	PAPDFFCS.pls
Body template	PAPDFFCB.pls
Package	pa_client_extn_dfftrans
Function	dff_map_segments_f
Procedure	dff_map_segments_PA_and_AP

Table 8 – 19 descriptive flexfield mapping extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Arguments Passed by the Calling Modules

The following table shows the arguments that the calling modules pass to the client extension.

The calling modules are:

- PRC: Interface Expense Reports from Payables (PAAPIMP)
- PRC: Interface Supplier Invoices from Payables (PAAPIMP)
- PRC: Interface Expense Reports to Payables (PATTER)
- PRC: Interface Supplier Invoice Adjustment Costs to Payables (PAVTVC)

The aliases used in the following table are:

- INV = AP_INVOICES
- DIST = AP_INVOICE_DISTRIBUTIONS
- CDL = PA_COST_DISTRIBUTION_LINES
- EI = PA_EXPENDITURE_ITEMS

Calling Module	p_trx_ref_1	p_trx_ref_2	p_trx_type	p_system_linkage_function	p_submodule
PAAPIMP (for supplier invoices)	DIST.invoice_id	DIST.distribution_line_number	INV.invoice_type_lookup_code	VI	INV.source
PAAPIMP (for expense reports)	DIST.invoice_id	DIST.distribution_line_number	'EXPENSE REPORT'	ER	INV.source
PATTER	EI.expenditure_item_id	CDL.line_num	'EXPENSE REPORT'	ER	NULL
PAVTVC	EI.expenditure_item_id	CDL.line_num	INV.invoice_type_lookup_code	VI	NULL

Table 8 – 20 Arguments passed by the calling modules (Page 1 of 1)

Sample Descriptive Flexfield Mapping Extension

The client extension body file, PAPDFFCB.pls, provides a sample descriptive flexfield mapping client extension. In the example, segments are mapped based on the system linkage function of the expenditure item.

DFF_Map_Segments_F Function

The dff_map_segments_f function provides the mapping logic for descriptive flexfields segments.

The default logic maps segment *n* in the originating application to segment *n* in the receiving application. You can change this function to map the segments according to your business rules.

This procedure uses the following parameters:

Parameter	Usage	Type	Description
p_attribute_number	IN	NUMBER	The identifier of the attribute to be mapped
p_calling_module	IN	VARCHAR2	The module that calls the extension
p_trx_ref_1	IN	NUMBER	Reference information passed to the extension
p_trx_ref_2	IN	NUMBER	Reference information passed to the extension
p_trx_type	IN	VARCHAR2	Type of transaction
p_system_linkage_function	IN	VARCHAR2	The expenditure type class function
p_submodule	IN	VARCHAR2	Name of the calling submodule
p_expenditure_type	IN	VARCHAR2	The expenditure type
p_set_of_books_id	IN	NUMBER	The set of books ID
p_org_id	IN	NUMBER	The organization ID
p_attribute_category	IN OUT	VARCHAR2	The context field value for the descriptive flexfield.
p_attribute_1 through p_attribute_10	IN OUT	VARCHAR2	The descriptive flexfield segment

Table 8 – 21 dff_map_segments_PA_to_AP parameters (Page 1 of 1)

DFF_Map_Segments_PA_and_AP Procedure

The dff_map_segments_PA_to_AP procedure calls the function dff_map_segments_f, and stores the mapped segments in the parameters p_attribute_1 through p_attribute_10.

You can modify this procedure to customize the attribute category mapping. An example of code for mapping the attribute category is provided in the extension.

This procedure uses the following parameters:

Parameter	Usage	Type	Description
p_calling_module	IN	VARCHAR2	The module that calls the extension
p_trx_ref_1	IN	NUMBER	Reference information passed to the extension
p_trx_ref_2	IN	NUMBER	Reference information passed to the extension
p_trx_type	IN	VARCHAR2	Type of transaction
p_system_linkage_function	IN	VARCHAR2	The expenditure type class function
p_submodule	IN	VARCHAR2	Name of the calling submodule
p_expenditure_type	IN	VARCHAR2	The expenditure type
p_set_of_books_id	IN	NUMBER	The set of books ID
p_org_id	IN	NUMBER	The organization ID
p_attribute_category	IN	VARCHAR2	
p_attribute_1 through p_attribute_10	IN	VARCHAR2	The descriptive flexfield segment number
x_status_code	OUT	VARCHAR2	Status of the procedure

Table 8 – 22 dff_map_segments_PA_and_AP parameters (Page 1 of 1)

See Also

Profile Options in Oracle Projects, *Oracle Projects Implementation Guide*

Archive Project Validation Extension

Use this extension to define additional business rules for validating projects. For a list of the basic business rules for validating projects see: Prerequisites for Purging Projects, *Oracle Projects Fundamentals.*

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PAXAPVXB.pls
Specification template	PAXAPVXS.pls
Package	PA_Purge_Extn_Validate
Procedure	Validate_Extn

Table 8 – 23 Pre–import extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Validate Projects Procedure

The Validate Projects procedure name is **Validate_Extn**. By default, the procedure returns NULL to the calling program.

Parameters

The validation extension uses the parameters shown in the following table.

Columns	Usage	Type	Constraint	Description
P_PROJECT_ID	IN	NUMBER	NOT NULL	Identifies the project to be purged
P_TXN_THROUGH_DATE	IN	DATE	NULL	For open projects, the date through which transactions are to be purged.
P_ACTIVE_FLAG	IN	VARCHAR2	NOT NULL	Indicates if the batch is created for open (active) projects
X_ERR_CODE	IN / OUT	NUMBER	NULL	Error handling code
X_ERR_STACK	IN / OUT	VARCHAR2	NULL	Error handling code
X_ERR_STAGE	IN / OUT	VARCHAR2	NULL	Error handling code. Default value is NULL.

Table 8 – 24 Archive Purge Validation Extension Parameters (Page 1 of 1)

Archive Custom Tables Extension

Use this extension if you want to archive and purge your custom tables. For example, if you maintain custom tables for project transaction data, you can use the Archive Custom Tables Extension to archive and purge these tables as part of the standard purge process.

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PAXAPPXB.pls
Specification template	PAXAPPXS.pls
Package	PA_Purge_Extn
Procedure	PA_Purge_Client_Extn

Table 8 – 25

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Archive Custom Tables Procedure

The Archive Custom Tables procedure name is **PA_Purge_Client_Extn**. By default, the procedure returns NULL to the calling program.

Parameters

The procedure uses the parameters shown in the following table.

Columns	Usage	Type	Constraint	Description
P_PURGE_BATCH_ID	IN	NUMBER	NOT NULL	Identifier of the purge batch
P_PROJECT_ID	IN	NUMBER	NOT NULL	Identifies the project to be purged

Columns	Usage	Type	Constraint	Description
P_PURGE_RELEASE	IN	VARCHAR2	NOT NULL	The Oracle Projects version used to run the purge
P_TXN_THROUGH_DATE	IN	DATE	NOT NULL	For open projects, the date through which transactions are to be purged
P_ARCHIVE_FLAG	IN	VARCHAR2	NOT NULL	Indicates if records in the custom table are to be archived.
P_CALLING_PLACE	IN	VARCHAR2	NOT NULL	BEFORE_PURGE or AFTER_PURGE indicates when the system calls the extension
P_COMMIT_SIZE	IN	NUMBER	NOT NULL	Number of archive and purge records processed before commitment
X_ERR_STACK	IN / OUT	VARCHAR2	NULL	Error handling code
X_ERR_STAGE	IN / OUT	VARCHAR2	NULL	Error handling code. The default value is NULL.
X_ERR_CODE	IN / OUT	NUMBER	NULL	Error handling code

Table 8 – 26 Purge Custom Tables Extension Parameters (Page 2 of 2)

CHAPTER

9

Oracle Project Costing Client Extensions

This chapter describes the client extensions in the Oracle Project Costing application.

Transaction Control Extensions

Transaction control extensions enable you to define your own rules to implement company–specific expenditure entry policies. Some examples of rules that you may define are:

- You cannot charge labor hours for a future date
- You cannot charge new transactions to projects for which the work is complete; you can only transfer items to these projects
- You can only charge to tasks that are managed by the organization you are assigned to
- All entertainment expenses are non–billable

See Also

Transaction Control Extensions Case Studies:

Case Study: New Charges Not Allowed: page 9 – 11

Case Study: Organization–Based Transaction Controls: page 9 – 13

Case Study: Default Billable Status by Expenditure type: page 9 – 15

Validation

You can use transaction control extensions to provide additional validation based on any type of data you enter in Oracle Projects. For example, you can check the project status for a particular project during expenditure entry.

You can validate any transaction entered into Oracle Projects, including transactions from other Oracle Applications and from external systems. For example, you can validate project–related supplier invoices entered into Oracle Payables. You can also validate items that you transfer from one project to another.

Transaction control extensions validate expenditures items one at a time; all validation is done for each expenditure item. Oracle Projects checks each expenditure item during data entry; the transaction is validated before you commit it to the database.

Processing

Oracle Projects processes transaction control extensions after the standard validation performed for expenditure entry, and after validating any transaction controls entered at the project or task level.

1. **Standard validation**
    - Transaction is within start and completion dates of project / task
    - Project status is not *Closed*
    - Task is chargeable
    - Transaction controls at project / task level
2. **Transaction control extension validation**

Designing Transaction Control Extensions

You should determine the logic and the additional data elements your client extensions require before you write them. We recommend that you consider some additional design issues for transaction control extensions:

- What are the business rules?
- What validation is required? Under what conditions does it apply?
- Are there any exceptions to the validation? How are exceptions handled?
- In what order should the transaction controls be executed if you have multiple rules?
- What error message should users see when entering a transaction not allowed by transaction control extensions?
- Are there any rules to set the default billable or capitalizable status of transactions?

See Also

Designing Client Extensions: page 7 – 6

Writing Transaction Control Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXTTCXB.pls
Specification template	PAXTTCXS.pls
Package	Patcx
Procedure	tc_extension

Table 9 – 1

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Writing Error Messages

You write error messages that will be displayed in forms when a transaction control violation is encountered. Use these messages to tell users why a particular transaction cannot be entered, based on validation in the procedure. These messages also appear on the Transaction Import exception report and indicate the reasons why transactions may be rejected by Transaction Import.

Be sure to define your messages under the Oracle Projects application. If you define your messages using the prefix **PATXC**, Oracle Projects will protect them during an upgrade.

The messages are stored in the table FND_NEW_MESSAGES.

See: Defining Messages *Oracle Application Object Library Reference Manual.*

Package.Procedure

patcx.tc_extension

The following table lists the parameters that Oracle Projects provides for the transaction control extension. All values are passed from the expenditure item being validated.

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	The identifier of the project.
X_task_id	IN	NUMBER	The identifier of the task.
X_expenditure_item_date	IN	DATE	The date of the expenditure item.
X_expenditure_type	IN	VARCHAR2	The type of expenditure.
X_non_labor_resource	IN	VARCHAR2	The non–labor resource; for usage items only.
X_incurred_by_person_id	IN	NUMBER	The identifier of the person incurring the transaction.
X_quantity	IN	NUMBER	The quantity of the transaction.
X_denom_currency_code	IN	VARCHAR2	The transaction currency code.
X_acct_currency_code	IN	VARCHAR2	The functional currency code.
X_denom_raw_cost	IN	NUMBER	The transaction currency raw cost.
X_acct_raw_cost	IN	NUMBER	The functional currency raw cost.
X_acct_rate_type	IN	VARCHAR2	The functional currency exchange rate type.
X_acct_rate_date	IN	DATE	The functional currency exchange rate date.
X_acct_exchange_rate	IN	NUMBER	The functional currency exchange rate.
X_transferred_from_id	IN	NUMBER	The identifier of the original expenditure item for which a new item is interfacing to a new project.
X_incurred_by_org_id	IN	NUMBER	The organization incurring the transaction.
X_nl_resource_org_id	IN	NUMBER	The identifier of the non–labor resource organization; for usages only.
X_transaction_source	IN	VARCHAR2	The transaction source of items imported using Transaction Import.
X_calling_module	IN	VARCHAR2	The module calling the extension.
x_vendor_id	IN	VARCHAR2	Identifier of the supplier

Table 9 – 2 (Page 1 of 2) Transaction Control Extension Parameters

Parameter	Usage	Type	Description
X_entered_by_user_id	IN	NUMBER	The identifier of the user that entered the transaction.
X_attribute_category	IN	VARCHAR2	Expenditure item descriptive flexfield context.
X_attribute1 through X_attibute–15	IN	VARCHAR2	Expenditure item descriptive flexfield segments.
X_msg_application	IN OUT	VARCHAR2	The application short name for the custom application providing customized messages
X_billable_flag	IN OUT	VARCHAR2	Determines whether or not a transaction is billable or capitalizable.
X_msg_type	OUT	VARCHAR2	Message type: W = warning message, E = error message.
X_msg_token1 through X_msg_token3	OUT	VARCHAR2	Message tokens used in warning messages.
X_msg_count	OUT	NUMBER	This parameter will support multiple messages in the future. In the current release, we support only one message, so the value is set to 1.
X_outcome	OUT	VARCHAR2	The outcome of the procedure.
P_projfunc_currency_code	IN	VARCHAR2	Identifier of the functional currency of the project–owning operating unit
P_projfunc_cost_rate_date	IN	VARCHAR2	Identifier of the exchange rate type used to convert the transaction cost amounts to the project functional currency
P_projfunc_cost_rate_date	IN	DATE	Identifier of the exchange rate date used to convert the transaction cost amounts to the project functional currency
P_projfunc_cost_exchg_rate	IN	NUMBER	Identifier of the exchange rate used to convert the transaction cost amounts to the project functional currency
x_assignment_id	IN	NUMBER	Identifier of the Project Resource Management assignment associated with the transaction
P_work_type_id	IN	NUMBER	Identifier of the work type assigned to the transaction
P_sys_link_function	IN	VARCHAR2	Expenditure type class of the transaction

Table 9 – 2 (Page 2 of 2) Transaction Control Extension Parameters

Additional Information About Parameters

Attributes

For the X_attribute parameters, you can use any attribute from the expenditure item descriptive flexfield. These parameters are not available for modules outside Oracle Projects.

Quantity

You can use the quantity parameter for validation using Oracle Projects and Oracle Payables features. However, keep in mind that Oracle Purchasing does not pass a value for this parameter.

Incurred by Person

Oracle Projects passes the person who is incurring the transaction. This value is always specified for labor and expense report items. It is optional for usage items, because you can enter usage logs which are incurred by an organization, and not an employee.

Oracle Payables passes a parameter value for supplier invoice transactions if the supplier of the invoice is an employee; otherwise this value is blank for supplier invoice transactions.

Oracle Purchasing does not pass a value for this parameter for requisitions and purchase orders transactions.

Billable/Capitalizable Flag

Oracle Projects passes in the billable value (contract projects) or capitalizable value (capital projects) that it has determined from the project and task transaction controls and the task billable status for this parameter. You can override this value based on logic that you write in your procedure. You can pass back a value of Y or N to specify the default billable or capitalizable status of a transaction. If you do not pass back a value, or if you pass back an invalid value, Oracle Projects uses the original value that it determined before calling the transaction control extension procedure.

Outcome Parameter

Use the X_outcome parameter to pass back the outcome of the procedure. If the transaction successfully passes all applicable

transaction control extension rules that you defined, leave the X_outcome parameter value as a null value. Oracle Projects then knows that this transaction passed all transaction control validation.

If the transaction does not pass a rule that you define, set the X_outcome value to the appropriate error message name that will be displayed to the user.

Calling Module

The calling module parameter indicates where the transaction control extension is being called from. You can base the logic of your extension on the calling module. For example, if Transaction Import is the calling module (PAXTRTRX), then allow only certain types of transactions to be charged to specific projects.

Below is a list of the possible values for the X_calling_module parameter. Note that these values are case–sensitive and are passed exactly as they appear.

When transaction controls is called by Oracle Purchasing and Oracle Payables, the validation is performed when you enter project–related information for requisitions, purchase orders, and supplier invoices. The validation is also performed when you enter or update the project–related information for distribution lines.

apiindib.pls	Payables invoice distributions
apiimptb.pls	Payables invoice import
APXINENT	Invoices Workbench in Oracle Payables. This value is passed when Transaction Controls is called to validate project–related information entered on a supplier invoice.
CreateRelated Item	CreateRelatedItem procedure called in the labor transactions extension procedure. This value is passed when CreateRelatedItem calls Transaction Controls to validate related transactions in the labor transactions extension procedure.
PAVVIT	Interface Supplier Invoices from Payables. This value is passed when Transaction Controls is called to validate expenditure items being created from project–related supplier invoice distribution lines interfaced from Oracle Payables into Oracle Projects.

PAXTREPE	Pre–Approved Expenditures. This value is passed when Transaction Controls is called to validate unapproved expenditure items being entered or updated in the Enter Pre–Approved Expense Reports form.
PAXTRTRX	Transaction Import. This value is passed when Transaction Controls is called by the Transaction Import program to validate transactions before they are loaded into Oracle Projects.
PAXEXCOP/ PAXTEXCB	Copy Pre–Approved Timecards / Copy Expenditures. This value is passed when Transaction Controls is called to validate new expenditure items being created using the Copy Pre–Approved Timecards feature.
PAXPRRPE	Adjust Project Expenditures. This value is passed when Transaction Controls is called to validate a new expenditure item that is being created as a result of an expenditure item transfer performed in the Adjust Project Expenditures form.
PAXVSSTS	Oracle Time and Labor
POWEBREQ	iProcurement
POXPOEPO	Purchase Orders in Oracle Purchasing. This value is passed when Transaction Controls is called to validate project–related information entered on a purchase order.
POXRQERQ	Requisitions in Oracle Purchasing. This value is passed when Transaction Controls is called to validate project–related information entered on a requisition.
POXPOERL	Releases in Oracle Purchasing. This value is passed when Transaction Controls is called to validate project–related information when you enter releases against purchase orders.
POXPOPRE	Preferences in Oracle Purchasing.
REQIMPORT	Requisition import
SelfService	Expense reports

Frequently Asked Questions

Can I Call Other Procedures within the Extension?

You can call other procedures. As long as you can determine the inputs and perform the validation for a particular rule, your extensions can be as flexible as you want them to be.

Can I Allow Exceptions to a Particular Rule?

Yes; for example, you can allow exceptions to a rule that applies to a project type by limiting the rule to particular projects for the project type in the procedure logic.

Can I Perform Validation on Groups of Expenditure Items?

Currently, you cannot perform validation on groups of expenditure items.

How Many Error Messages Can My Procedure Return?

Your procedure can return one error message, which is the first error message that Oracle Projects encounters in your procedure.

See Also

Case Study: New Charges Not Allowed: page 9 – 11

Case Study: Organization–Based Transaction Controls: page 9 – 13

Case Study: Default Billable Status by Expenditure type: page 9 – 15

Case Study: New Charges Not Allowed

This case study demonstrates how to use a client extension to disallow new charges to completed projects.

Business Rule

You have decided that you do not want anyone to charge new transactions to projects for which the work is complete. However, to properly account for project work performed, these projects will allow new transactions resulting from transfers between projects.

Requirements

The business rule will be carried out as follows:

- Do not allow new expenditure items to be charged to projects having a project status of *Processing Only*
- Allow expenditure items to be transferred to projects having a project status of *Processing Only*
- Display an error message when a user tries to enter new expenditure items charged to projects having a project status of *Processing Only*
- Do not allow any exceptions to this business rule

You could easily implement an exception to this rule regarding new charges from transfers only. An exception to this rule is to also allow supplier invoice transactions, which are typically received after the project work is complete.

Required Extension

To implement the business rule of controlling new charges to projects for which the work is complete, use the transaction control extension.

Suggestion: Review the sample PL/SQL code that corresponds to the implementation of this case study in the file PAXTTCXB.pls.

Additional Implementation Data

You need to define a new project status of *Processing Only*.

Design Considerations for New Charges Not Allowed

The design considerations are described below:

Identifying Transferred Items

You know if the item you are validating is a transfer from another project or task by looking at the value of the x_transferred_from_id parameter passed into your extension.

Determining Project Status

The project status is not passed as a parameter to the transaction control extension. Therefore, you need to derive this value from the project ID.

Defining an Error Message

If an item is a new item being charged to a project with a project type having the status of *Processing Only*, you want to display an error message to the user. The user can then change the project assignment of the new expenditure item to a different project.

You define an error message with the text, ”You cannot create new items for *Processing Only* projects”.

Case Study: Organization–Based Transaction Controls

This case study demonstrates how to use a client extension to set up transaction controls by organization.

Business Rule

You want all administrative work to be charged to tasks that are managed by the employee's organization. When the employee is not specified, charge the administrative work to the expenditure organization.

Requirements

The business rule will be carried out like this:

- Tasks with a service type of *Administration* allow charges only for employees assigned to the same organization as the task–owning organization
- For usages not associated with a specific employee, the expenditure item must have been charged by the same expenditure organization as the task organization
- Display an error message when a user tries to enter an expenditure item that violates this rule
- Do not allow any exceptions to this business rule

You can easily implement an exception to this rule, in which this rule does not apply to any projects that are managed by the *Executive* office. This exception exists because the Executive office uses resources throughout the company to perform important administrative work. The Executive office does not want to set up projects with a task for every organization that may help with the project work.

Required Extension

To implement the business rule of organization–based transaction controls, use the **Transaction Control Extension**.

Suggestion: Review the sample PL/SQL code that corresponds to the implementation of this case study in the file PAXTTCXB.pls.

Additional Implementation Data

You need to define a new task service type of *Administration*.

Design Considerations for Organization–Based Transaction Controls

The design considerations are described below:

Determining Incurred by Organization

Since the incurred by organization of each transaction being evaluated is passed to the transaction control extension procedure, you do not need to derive the organization.

Determining Task Organization

Task organization is not passed as a parameter to the transaction control extension. Therefore, you need to derive this value.

Determining Task Service Type

The task service type is not passed as a parameter to the transaction control extension. Therefore, you need to derive this value.

Defining an Error Message

If an item being charged to a task violates this rule, you want to display an error message to the user. The user can then change the task assignment to a different value.

You define an error message with the text, ”Only the task–owning organization can charge to this task”.

Case Study: Default Billable Status by Expenditure Type

This case study demonstrates how to use a client extension to specify a default billable status based on the expenditure type.

Business Rule

You have decided that you want to implement the business rule that no one can bill entertainment charge to projects.

Requirements

The business rule will be carried out like this:

- Transactions with an expenditure type of *Entertainment* are non–billable for all projects, regardless of the task's billable status
- There are no exceptions to this rule within the client extension; exceptions for negotiated billing of *Entertainment* expenses are marked as billable using the Adjust Project Expenditures form.
- Do not return an error message to the user for any expenditure types of *Entertainment*; simply set the billable status to non–billable for affected transactions.

Required Extension

To implement the business rule of determining the default billable status by expenditure type, use the **Transaction Control Extension**.

Suggestion: Review the sample PL/SQL code that corresponds to the implementation of this case study, view the file PAXTTCXB.pls.

Additional Implementation Data

You need to define a new expenditure type of *Entertainment*.

Design Considerations for Default Billable Status by Expenditure Type

The design considerations are described below:

Deriving Additional Information

Since the expenditure type of each transaction being evaluated is passed to the transaction control extension procedure, you do not need to derive any additional data to implement this business rule.

Determining Billable Status

You can simply code your procedure to look at the expenditure_type parameter; if the expenditure type is Entertainment, set the x_billable_flag parameter to N to implement this business rule.

AutoApproval Extensions

The AutoApproval Extensions contain procedures to define conditions under which expense reports and timecards are approved automatically.

Each procedure includes examples that you can copy and modify. The AutoApproval extensions include the following procedures:

Body Template	Specification Template	Package	Procedure
PAXPTEEB.pls	PAXPTEES.pls	pa_client_extn_pte	get_exp_autapproval This procedure contains default logic to read the values of the AutoApproval profile options.
PAXTGTCB.pls	PAXTGTCS.pls	pagtcx	summary_validation_extension This procedure performs custom validation for all the expenditure items in an expenditure.
PAXTRT1B.pls	PAXTRT1S.pls	pa_client_extn_rte	check_approval Use this procedure to incorporate additional approval logic for timecards.
PAXTRTEB.pls	PAXTRTES.pls	paroutingx	route_to_extension Use this procedure to define rules for routing timecards and expense reports for approval.
PAPSSTCB.pls	PAPSSTCS.pls	pa_time_client_extn	display_business_message Use this procedure to define validations during entering and approval of timecards in Oracle Time and Labor.

Table 9 – 3 AutoApproval Procedure Parameters (Page 1 of 1)

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

AutoApproval Parameters

The AutoApproval Extension procedures use the following parameters:

Parameter	Usage	Type	Description
X_source	IN	VARCHAR2	Identifies the source of the expenditure
X_exp_class_code	IN	VARCHAR2	Identifies the expenditure class (OT for timecards and OE for expense reports)
X_txn_id	IN	NUMBER	System–generated identifier of the expenditure (passed in by the form). For expenditures created in Oracle Projects, this is the expenditure ID.
X_exp_ending_date	IN	DATE	Ending date of the expenditure week
X_person_id	IN	NUMBER	The values of Incurred_By_Person_Id for the timecard (when Oracle Internet Time is used).
P_module	IN	VARCHAR2	Identifies the module calling the procedure (for example, Self Service Time and Oracle Time and Labor).
P_timecard_table	IN	PL/SQL table	Table of expenditure items included on the timecard.
X_approved	IN / OUT	VARCHAR2	Value of the AutoApproval profile option

Table 9 – 4 AutoApproval Procedure Parameters (Page 1 of 1)

For more information about this client extension, see the *Oracle Time and Labor System Administrator's Guide*.

Labor Costing Extensions

Labor costing extensions allow you to derive raw cost amounts for individual labor transactions. Some examples of labor costing extensions you may define are:

- Standard cost rate by job
- Capped labor cost rates
- Multiple cost rates per employee

You can use labor costing extensions to implement unique costing methods other than the standard method, which calculates raw cost using the number of hours multiplied by the employee's hourly cost rate. For example, you may want to calculate the raw cost using a capped labor rate for specific employees.

Processing

Oracle Projects processes labor costing extensions during labor cost distribution before calculating standard raw cost amounts. If Oracle Projects encounters a labor costing extension that derives the raw cost amount of a labor transaction, it skips the standard raw cost calculation section for that transaction.

Designing Labor Costing Extensions

Consider the following design issues for labor costing extensions:

- What are the conditions and circumstances in which you cannot use the standard raw cost calculation method supported by Oracle Projects?
- How is the raw cost amount calculated in these cases?
- How do you identify labor transactions that meet these conditions?
- How do you store rates and other additional information that your calculations may require? How are the rates and other information maintained?
- What are the exception conditions for your labor costing extension? What is the exception handling if you cannot find a rate that should exist?

The extension is identified by the following items:

Item	Name
Body template	PAXCCECB.pls
Specification template	PAXCCECS.pls
Package	PA_Client_Extn_Costing
Procedure	Calc_Raw_Cost

Table 9 – 5

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Package.Procedure

PA_Client_Extn_Costing.Calc_Raw_Cost

The following table lists the parameters that Oracle Projects provides for the labor costing procedure.

Parameter	Usage	Type	Description
x_transaction_type	IN	VARCHAR2	An idenfier that distinguishes between actual and forecast transactions. This enables you to process forecast and actual transactions differentlyThe default value is ACTUAL. Forecast transactions can have the following transaction types: ROLE ASSIGNMENT
x_expenditure_item_id	IN	NUMBER	The identifier of the expenditure item.

Table 9 – 6 (Page 1 of 2) Labor Costing Extension Parameters

Parameter	Usage	Type	Description
x_sys_linkage_function	IN	VARCHAR2	The expenditure type class of the expenditure item.
x_denom_raw_cost	IN OUT	NUMBER	The raw cost amount.
x_status	IN OUT	NUMBER	The status of the procedure.

Table 9 – 6 (Page 2 of 2) Labor Costing Extension Parameters

Using Raw Cost

The raw cost amount that your procedure calculates is assigned to the x_raw_cost parameter. Leave this value blank if you want to use the standard costing method which uses the employee's hourly cost rate.

If you pass a value to this parameter, Oracle Projects calculates the raw cost rate of the transaction using the x_raw_cost parameter value divided by the number of hours.

Using Status

Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

x_status = 0 The extension executed successfully.

x_status < 0 An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.

x_status > 0 An application error occurred. Oracle Projects writes a rejection reason to PA_EXPENDITURE_ITEMS.COST_DIST_REJECTION_CODE and does not cost the transaction. You can review the rejection reason in the labor cost distribution exception report.

Labor Transaction Extensions

Labor transaction extensions allow you to create additional transactions for individual labor items charged to projects. For example, you may wish to create additional transactions for hazardous work performed for every labor transaction charged to certain projects. Here are some other examples of labor transactions extensions you can implement:

- Create overtime premium transactions for overtime hours based on company overtime policies
- Create fringe benefit transactions which are charged to the same project the source labor was charged to

You can create additional transactions for straight time labor transactions and overtime labor transactions. You create additional labor transactions based on the source labor transactions that you enter on timecards.

Related Transactions

Additional transactions that are created for labor transactions are referred to as *related transactions*. All related transactions are associated with a *source transaction* and are attached to the expenditure item ID of the source transaction. You can identify and process the related transactions by referring to the expenditure item ID of the source transaction.

You create related transactions to process a raw cost amount separately than the source transaction raw cost amount. Related transactions can be burdened, billed, and accounted for independently of the source transaction.

Processing

Oracle Projects processes labor transaction extensions during labor cost distribution. When you distribute labor costs, the labor transaction extension is processed after the raw cost calculation of the source transactions. This allows you to derive the cost of the related transaction from the cost of the source transaction.

You also use the labor transaction extension to calculate new cost amounts for related transactions if the source transaction is recosted.

If you are using the Labor Transaction Extension to create overtime premium transactions, you may not need to use the Overtime Calculation program that Oracle Projects provides. If you determine that you need to use both the Labor Transaction Extension and the Overtime Calculation program, you need to ensure that you have defined conditions so that each transaction is processed by only one of these processes, based on your company policies.

See Also

Distributing Labor Costs, *Oracle Project Costing User Guide*

Creating Overtime, *Oracle Project Costing User Guide*

Adjustments to Related Transactions, *Oracle Project Costing User Guide*

Designing Labor Transaction Extensions

Consider the following design issues for labor transaction extensions:

- What are the conditions in which your company needs to create related items? Why are you creating related items instead of using another method like burdening to account for additional costs?
- How do you identify labor transactions that meet these conditions?
- What related transactions should be created in these cases?
- What project and task are the related transactions charged to?
- What expenditure types are used for the related transactions?
- How is the raw cost of the related transaction calculated? Is it based on the raw cost of the source transaction or based on some other calculation?
- Is the related transaction burdened? If so, you need to set up your cost plus implementation so that the transaction is burdened.
- How is the related transaction's cost accounted for? Is the raw cost accounting for related transactions different from the accounting for source transactions? Is the total burdened cost

accounting different (if you use total burdened cost accounting)? You need to define your AutoAccounting rules for labor costs appropriately.

- How is the billable status of each related transaction determined? Do you need to create a transaction control extension rule to properly specify the related transaction's billable status?
- Are the related transactions billed? If so, under what conditions? How is the bill amount calculated under the different billing methods? Do you need to use a labor billing extension to bill these transactions?
- Is the related transaction's revenue accounted for differently than the source transactions? If so, how? You need to define your AutoAccounting rules for labor revenue appropriately.
- What are the exception conditions for your labor transaction extension? For example, what is the exception handling if you cannot find a rate for the related transaction if the related transaction's raw cost is not directly based on the source transaction's raw cost?

See Also

Designing Client Extensions: page 7 – 6

Writing Labor Transaction Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXCCETB.pls
Specification template	PAXCCETS.pls
Package	PA_Client_Extn_Txn
Procedure	Add_Transactions

Table 9 – 7

Oracle Projects also provides two public procedures that you use within the Add_Transactions procedure for the following purposes:

- Creating Related Transactions
- Updating Related Transactions

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Adding Transactions

Use the Add_Transactions procedure to add related transactions for source transactions. Within this procedure, you write logic to create related new transactions and update the raw cost of related transactions when they are marked for cost recalculation. You calculate the raw cost of related transactions in this procedure only; Oracle Projects does not calculate the raw cost of related transactions in any other way. Use the two procedures discussed later in this section for processing related transactions within this procedure.

PA_Client_Extn_Txn.Add_Transactions

The following table lists the parameters that Oracle Projects provides for the add related transactions procedure.

Parameter	Usage	Type	Description
x_expenditure_item_id	IN	NUMBER	The identifier of the source transaction.
x_sys_linkage_function	IN	VARCHAR2	The expenditure type class of the source transaction.
x_status	IN OUT	NUMBER	The status of the procedure.

Table 9 – 8 (Page 1 of 1) Add Related Transactions Parameters

Creating Related Transactions

Use this procedure to create related transactions within the logic of the Add Transactions procedure. This procedure exists in the **pa_transactions** package; you cannot change this procedure.

The related transaction is linked to the same employee's timecard as the source transaction. The transaction is created with a quantity of 0, in order to maintain the proper number of hours for the employee's timecard, even when related transactions exist.

pa_transactions.CreateRelatedItem

The CreateRelatedItem procedure does the following:

- Ensures all input parameter values are valid values
- Ensures that the expenditure type is classified with an expenditure type class of *Straight Time* or *Overtime*
- Validates that the transaction passes all transaction controls validation rules, including logic in transaction control extensions
- Determines the billable status of the related transaction using the same method used for all Oracle Projects transactions
- If the transaction is valid, creates related labor expenditure item that:
  - Is attached to the source transaction's expenditure
  - Has quantity of 0 (to maintain the number of hours for the employee's timecard, even when related items exist for that timecard)
  - Uses the source transaction's project and task unless you specify project and task input values
  - Uses the source transaction's expenditure item date and bill hold value
  - Uses the source transaction's organization unless you specify an override organization
  - Rounds the raw cost to 2 decimal places and uses the raw cost rate that you passed into it

The following table lists the parameters that Oracle Projects provides for the create related transactions procedure.

Parameter	Usage	Type	Description
x_source_exp_item_id	IN	NUMBER	The identifier of the source transaction.
x_project_id	IN	NUMBER	The identifier of the project to charge the related transaction to.
x_task_id	IN	NUMBER	The identifier of the task.
x_expenditure_type	IN	VARCHAR2	The expenditure type of the related transaction.
x_raw_cost	IN	NUMBER	The raw cost amount of the related transaction.
x_raw_cost_rate	IN	NUMBER	The raw cost rate of the related transaction.
x_override_to_org_id	IN	NUMBER	The identifier of the organization that overrides the expenditure organization used by the source transaction.
x_userid	IN	NUMBER	The identifier of the user that entered the source transaction.
x_work_type_name	IN	VARCHAR2	Name of the work type assigned to the transaction
x_attribute_category	IN	VARCHAR2	Descriptive flexfield context.
x_attribute1 – 10	IN	VARCHAR2	Descriptive flexfield segments.
x_comment	IN	VARCHAR2	Expenditure item comment.
x_status	OUT	NUMBER	Status of the procedure.
x_outcome	OUT	VARCHAR2	Outcome of the procedure.

Table 9 – 9 (Page 1 of 1) Create Related Item Parameters

Updating Related Transactions

Use this procedure to update the raw cost amount of existing related transactions within the logic of your labor transaction extension when

related transactions are marked for cost recalculation. This procedure exists in the **pa_transactions** package; you cannot change this procedure.

pa_transactions.UpdateRelatedItem

The following table lists the parameters that Oracle Projects provides for the update related transactions procedure.

Parameter	Usage	Type	Description
x_expenditure_item_id	IN	NUMBER	The identifier of the related expenditure item.
x_raw_cost	IN	NUMBER	The new raw cost of the related transaction.
x_raw_cost_rate	IN	NUMBER	The new raw cost rate of the related transaction.
x_status	OUT	NUMBER	Status of the procedure.

Table 9 – 10 (Page 1 of 1) Update Related Item Parameters

Additional Information About Parameters

Additional details about some of the parameters is shown below.

Using Project and Task in the CreateRelatedItem Procedure

You can optionally pass the project and task parameter values to the CreateRelatedItem procedure.

If you do not pass project and task information, Oracle Projects charges the related transaction to the same project and task that the source transaction is charged to.

If you do pass project and task information, Oracle Projects uses these values to ensure that the transaction can be charged based on the transaction control validation for that project and task. If the related transaction passes all transaction control rules, then the related transaction is created with that project and task. You must pass both a

project and task value to override the source transaction's project and task.

Using Userid in the CreateRelatedItem procedure

You must provide an input value for the X_userid parameter for the CreateRelatedItem procedure. Oracle Projects passes this value to the transaction control procedure, which is called before the related transaction is created. You may have defined logic in your transaction control extensions that uses the userid value. You typically pass the user of the person who created the source transaction, but you can pass any userid that you want to the CreateRelatedItem procedure.

Using an Override Organization in CreateRelatedItem Procedure

Use the x_override_to_org_id to override the source transaction's expenditure organization to another organization, such as the project organization for the related transaction.

If a value is provided for this parameter when calling the create related transactions procedure, and it is a valid organization, then that value is stored as the expenditure item's override organization *regardless* of the existence of any other cost distribution overrides defined for the project.

This organization is then used when calculating burdened amounts for the related transaction. It is also used as the input value for any AutoAccounting rules that use the expenditure organization parameter.

However, the source transaction expenditure organization is what the create related transaction procedure passes to the transaction controls procedure for validation. This is done to retain consistency with expenditure entry forms which always send the incurred by (or expenditure organization) organization value. The expenditure organization parameter is used in Transaction Control Extensions by clients who want to control expenditure entry by what organization is *charging to* the project.

Therefore, the override organization value is used only for burdening and AutoAccounting.

Using Outcome in the CreateRelatedItem Procedure

Oracle Projects uses the X_outcome parameter to pass back the rejection reason encountered in the application logic of the CreateRelatedItem procedure. For example, if the related transaction

is rejected by the transaction controls validation called in the CreateRelatedItem procedure, then the reason is assigned to the X_outcome parameter.

Using Status in both Procedures

Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

x_status = 0 The extension executed successfully.

x_status < 0 An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.

x_status > 0 An application error occurred. Oracle Projects writes a rejection reason to PA_EXPENDITURE_ITEMS.COST_DIST_REJECTION_CODE and does not cost the source and related transactions. You can review the rejection reason in the labor cost distribution exception report.

The two related transaction procedures pass your labor transaction procedure the outcome of their processing in this same way as you pass the outcome of your labor transaction extension procedure to the labor distribution process.

Frequently Asked Questions

Following are some frequently asked questions about the labor transaction extension.

What Happens if the Source Transaction is Not Costed?

If the source transaction is not costed because it is rejected during cost distribution, the labor transaction extension is not called for that transaction. Therefore, related transactions for rejected source transactions will not be created or costed.

Can I Create Multiple Related Transactions for a Single Item?

Yes, you can create multiple related transactions for a given source transaction based on the logic in your labor transaction extension.

How Do I Identify Related Transactions?

You identify related transactions by referring to the expenditure item id of the source transaction.

In the expenditure inquiry forms and reports within Oracle Projects, you can identify related transactions based on your implementation data used for related transactions, particularly the expenditure type. Oracle Projects displays all related transactions immediately after the source transaction.

What if Some Parameters Are Not Passed to CreateRelatedItem?

All parameters that are not passed to the related transactions procedure are read from the source transaction; except for quantity, billable status, and expenditure type. The quantity is set to 0 for the related transactions. The billable status is derived based on the transaction controls and transaction control extensions that you define. Expenditure type is a required parameter that you provide.

What if a Related Transactions Does Not Pass Validation?

If a related transaction does not pass validation in the CreateRelatedItem procedure, Oracle Projects does not create the related item, and marks the source transaction with a cost distribution rejection reason specifying that an error was encountered in the labor transaction extension procedure. The source item is not marked as cost distributed and is displayed in the exception output report in the Distribute Labor Costs process.

Where Can I Establish the Billable or Capitalizable Status of Related Transactions?

The related transaction's billable or capitalizable status is derived using transaction controls and task billable or capitalizable status like all other transactions. You can further derive the billable or capitalizable status of related transactions by including logic in the transaction control extension procedure to look at related transactions based on certain criteria, and then setting the billable or capitalizable flag. The

transaction control package, which establishes the billable or capitalizable status, is called within the CreateRelatedItem procedure.

How Does the Transaction Controls Procedure Identify Related Transactions?

The transaction control procedure, which establishes the billable or capitalizable status and validates transactions, is called within the CreateRelatedItem procedure.

The transaction control extension identifies related transactions by the x_module of the CreateRelatedItem procedure. When the calling procedure (CreateRelatedItems) calls transaction controls, the x_module is set to *CreateRelatedItem*.

Can I Calculate Raw Cost Amounts of Related Transactions Using Burden Costing?

You can use the Cost Plus API to determine raw cost amounts of related transactions based on your burden costing setup.

See Also

Cost Plus Application Programming Interface (API): page 4 – 15

Adjusting Related Transactions

Whenever an adjustment is performed on a source transaction that requires the item to be backed out (transfer, split, manual reversal through the Pre–Approved Expenditure form), Oracle Projects creates reversals for the related transactions of the source transaction.

You cannot independently process related transactions from the source transactions. However, there are adjustment actions for which related transactions are processed with the source transaction.

See Also

Adjustments to Related Transactions, *Oracle Project Billing User Guide*

Overtime Calculation Extension

The overtime calculation extension allows you to define your own rules to implement company–specific overtime calculation policies. The extension calculates overtime costs and charges them to an indirect project other than the project where the labor was charged.

> **Note:** If you want to charge overtime to the project where the labor was charged, consider creating items via the labor transaction extension. See: Labor Transaction Extensions: page 9 – 22.

For more information on the context and setup of overtime calculations, see: Implementing Overtime Charged to an Indirect Project, *Oracle Projects Implementation Guide*.

Processing

Oracle Projects calls the Overtime Calculation Extension during the Distribute Labor Costs process.

See Also

Overview of Tracking Overtime, *Oracle Project Costing User Guide*

Designing Overtime Calculation Extensions

Oracle Projects provides a template Overtime Calculation extension. You can use the template to understand the extension, and then make appropriate changes to meet your business needs. Before modifying the extension, read the following essay and related case studies on implementing overtime: Overview of Tracking Overtime, *Oracle Project Costing User Guide*.

Implementing Your Company's Overtime Calculation Extension

If you decide to use automatic overtime calculation, you can implement your company's overtime policies using the template Overtime Calculation extension as a starting point.

Your technical staff can customize the Overtime Calculation extension to accommodate the overtime rules that your business uses.

We recommend that you complete the following steps to implement your company's Overtime Calculation extension:

- Define and document your overtime policy
- Use your documented overtime policy to determine the kind of implementation data you need to drive automatic overtime calculation. This implementation data may include labor costing rules, expenditure types, labor cost multipliers, and an overtime project and tasks
- Define the implementation data necessary to drive automatic overtime calculation
- Have your technical staff code your overtime policy in the Overtime Calculation extension
- Test your implementation data and Overtime Calculation extension to ensure that it correctly implements your company's overtime policies

A few additional notes about implementing the Overtime Calculation extension are:

- Define all overtime expenditure types with an end date so that timecard clerks cannot enter overtime through the Pre–Approved Expenditures window
- Base automatic overtime calculation on weekly overtime rules. Oracle Projects is designed to process weekly timecards; all expenditure item dates of a timecard must be within the expenditure week ending date of the timecard. Therefore, automatic overtime calculation is most easily performed based on weekly overtime rules

How the Overtime Calculation Extension Processes Overtime

The Overtime Calculation extension template follows these steps to process overtime:

- Determines all employees and corresponding weeks which may include new overtime to process. The Overtime Calculation extension calculates and creates overtime only for employees with timecards processed in the run of Distribute Labor Costs that calls the Overtime Calculation extension. These employees and weeks are identified by the request_id of the straight time

expenditure items that are costed before the Overtime Calculation extension is called.

- Sums the hours required to calculate overtime for identified employees and weeks. The standard Overtime Calculation extension sums the total hours for the week and the total hours for each day of the week, relying on the timecard entry validation rule that all labor expenditure item dates must be within the expenditure week ending date of the timecard.
- Calculates overtime hours based on the hours worked, the employee's labor costing rule, and other criteria you might specify. The standard Overtime Calculation extension calculates overtime for an employee and a week based on the employee's labor costing rule described in the case study. See: Implementing Overtime Charged to an Indirect Project, *Oracle Projects Implementation Guide*.
- Creates overtime expenditure items for each type of overtime for which the employee is eligible. The overtime item is charged to the overtime project and appropriate overtime task that is specified in the Overtime Calculation extension using the overtime expenditure type defined for the employee's labor costing rule. The expenditure item date is set to the week ending date. The expenditure item is assigned the labor cost multiplier that is associated with the overtime task to which it is charged.

  The extension creates a new expenditure for each person and week that has new overtime items. The new expenditures are assigned to an expenditure batch created in the Overtime Calculation extension. The expenditure batch name is based on the Request ID number, and uses the prefix "PREMIUM". For example, an expenditure batch run under Request ID 1205 would be named *PREMIUM – 1205*.
- Lists employees with new overtime items on the Overtime Calculation Report.

After the Overtime Calculation extension has completed, the Distribute Labor Costs process costs the new overtime items.

Writing the Overtime Calculation Extension

The extension is identified by the following items:

Item	Name
Body template	PAXDLCOB.pls
Specification template	PAXDLCOS.pls
Package	pa_calc_overtime

Table 9 – 11

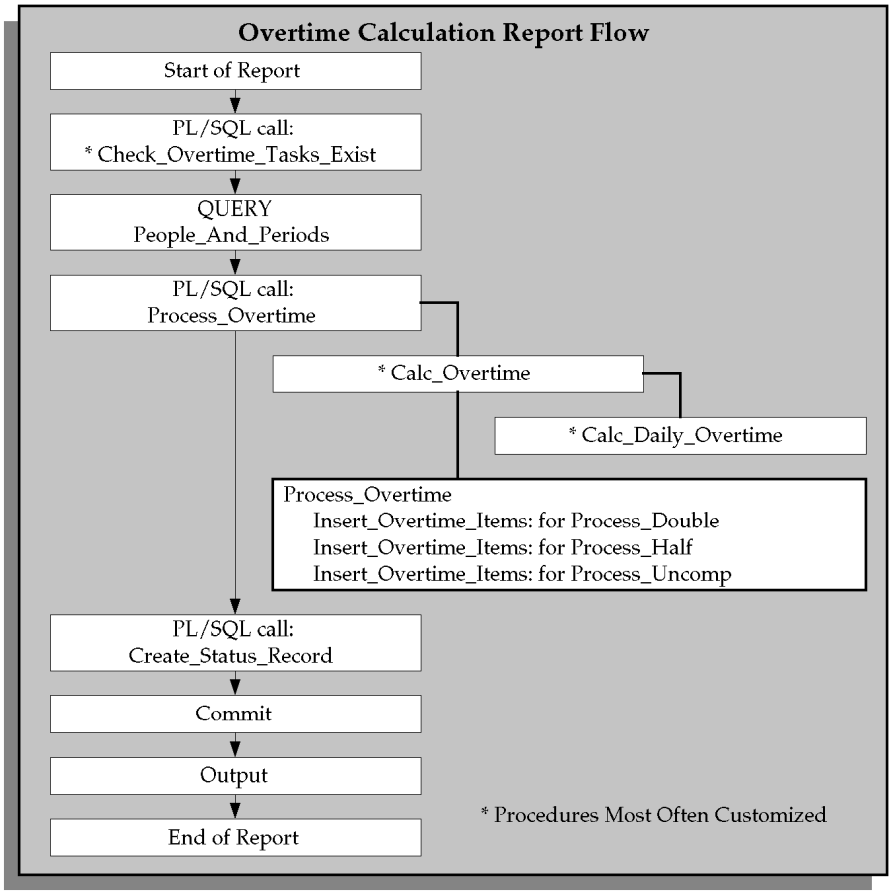
Structure of the Overtime Calculation Report

The Overtime Calculation Report is an output report generated by the Distribute Labor Costs process, using procedures in the Overtime Calculation extension. The report is only generated if you have implemented the Overtime Calculation extension.

The name of the template report is PAXDLIOT.rdf. It is located in the Oracle Projects reports directory. You do not need to modify this report. You should only need to modify the PL/SQL procedures in the overtime calculation extension template package. See: Writing the Overtime Calculation Extension: page 9 – 36.

Figure 9 – 1 shows the structure of the Overtime Calculation Report. The procedures you are most likely to modify to implement your company's overtime rules are marked with an asterisk (*) in the diagram.

Figure 9 – 1



The report first calls the Check_Overtime_Tasks_Exist procedure. This procedure looks for overtime projects and tasks and returns all relevant task names, up to a maximum of five. These tasks determine the column titles in the report.

Next, the report queries the database for all records processed by the Distribute Labor Costs process. The report then calls the Process_Overtime procedure. This procedure determines the amount and type of overtime for each employee and period, creates new expenditure items for these values, and passes the values back to the report.

Calc_Overtime and Calc_Daily_Overtime are procedures used by the Process_Overtime procedure. You can decide whether to use these procedures in your customized extension.

Your extension must also adjust overtime that relates to any adjustments made to the original transactions. For best results, use the Process_Overtime procedure to create the new overtime records, as this procedure handles all the inserts and updates to the Oracle Projects tables.

Finally, the report calls the Create_Status_Record procedure. This procedure is called in the report PAXDLCOT.rdf to create a status record for the overtime calculation program. This record lets the costing program know whether the overtime calculation program is complete.

Burden Costing Extension

Use the Burden Costing client extension to override the burden schedule ID.

Oracle Projects calls the Burden Costing extension during the cost distribution processes. You can modify the extension to satisfy your business rules for assigning burden schedules.

Description

The extension is identified by the following items:

Item	Name
Specification template	PAXCCEBS.pls
Body template	PAXCCEBB.pls
Package	pa_client_extn_burden
Procedure	override_rate_rev_id

Table 9 – 12 Burden Costing extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Override_Rate_Rev_ID Procedure

The override_rate_rev_id procedure assigns a burden cost schedule to a transaction. This procedure uses the following parameters:

Parameter	Usage	Type	Description
p_transaction_type	IN	VARCHAR2	An idenfier that distinguishes between actual and forecast transactions. This enables you to process forecast and actual transactions differently The default value is ACTUAL. Forecast transactions can have the following transaction types: ROLE ASSIGNMENT
p_tran_item_id	IN	NUMBER	The identifier of the transaction
p_tran_type	IN	VARCHAR2	The transaction type
p_task_id	IN	NUMBER	The task ID
p_schedule_type	IN	VARCHAR2	The rate schedule type: C = costing schedule R = revenue schedule I = invoice schedule
p_exp_item_date	IN	DATE	The expenditure item date
x_sch_fixed_date	OUT	DATE	The schedule fixed date for firm costing schedules
x_rate_sch_rev_id	OUT	NUMBER	The burden schedule revision ID assigned by the extension
x_status	OUT	NUMBER	Status of the procedure: 0 = successful execution <0 = Oracle error >0 = application error

Table 9 – 13 override_rate_rev_id parameters (Page 1 of 1)

See Also

Entering Project and Task Options, *Oracle Projects Fundamentals*

Rate Schedules, *Oracle Projects Implementation Guide*

Allocation Extensions

You can use the allocation extensions to expand the capabilities of the allocations feature.

Each allocation extension includes examples that you can copy and modify.

The allocations extensions include:

- Allocation Source Extension: page 9 – 41
- Allocation Target Extension: page 9 – 43
- Allocation Offset Tasks Extension: page 9 – 46
- Allocation Offset Projects and Tasks Extension: page 9 – 47
- Allocation Basis Extension: page 9 – 49
- Allocation Descriptive Flexfields Extension: page 9 – 50
- Allocation Dependencies Extension: page 9 – 52

See Also

Allocations, *Oracle Project Costing User Guide*

Allocation Source Extension

This extension defines source projects and tasks. Oracle Projects calls this procedure when Use Client Extension Sources is selected in the Source window.

Use the Allocation Source extension when you want to include or exclude projects or tasks temporarily when creating a source pool. You may also find that it is more convenient to maintain a large list of source projects in the extension file rather than in the Sources window.

Description

For each allocation rule_id, the client populates the global session variable x_source_proj_tasks_tbl of the data type table alloc_source_tabtype. The allocation run process reads this table and uses the projects and tasks as the sources for any allocation run that

uses the rule. The projects and tasks are added to those projects and tasks specified in the source lines.

The extension includes the following items:

Item	Name
Body template	PAPALCCB.pls
Specification template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	source_extn

Table 9 – 14 Allocation Source Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_alloc_rule_id	IN	Number	Identifies the allocation rule
x_source_proj_task_tbl	OUT	alloc_source_tabtype	Number the index sequentially from 1. Otherwise the process will fail.
x_error_message	OUT	VARCHAR2(30)	Error message text
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error

Table 9 – 15 Allocation Source Extension Parameters (Page 1 of 1)

Additional Parameter Information

The datatype *alloc_source_tabtype* contains the following parameters:

Parameter	Type	Description
project_id	Number	(Required) Identifies the source project. The source project and the allocation rule must be from the same operating unit.
task_id	Number	Identifies the source task (must be a top or lowest task)
exclude_flag	Varchar2(1)	(Default is N) If Y, exclude the project and task from the source project and tasks

Table 9 – 16 Additional Parameters: Allocation Source Extension (Page 1 of 1)

Validation

The Generate Allocation Transactions process:

- Validates project_id against the single organization view pa_projects
- Verifies that the project is open (that is, pa_project_stus_utils.is_project_closed(project_id) ='N' and template_flag ='Y')
- Validates task_id against view pa_alloc_src_tasks_v
- Verifies that the task belongs to the source project

If the validation fails, the Generate Allocation Transactions process populates the message "The client extension returned an invalid project or task."

Allocation Target Extension

This extension defines target projects and tasks. Oracle Projects calls this extension when Use Client Extension Targets is selected in the Targets window.

Use the Allocation Targets extension when you want to include or exclude projects or tasks temporarily when allocating amounts to target projects and tasks. You may also find that it is more convenient to maintain a large list of target projects in the extension file rather than in the Targets window.

Description

For each allocation rule_id, the client populates the global session variable x_target_proj_task_tbl of the data type table alloc_target_tabtype. The allocation run process reads the table and uses the specified project and chargeable tasks as the target for the allocation run. The system can use both the projects and tasks specified in the extension as well as those specified on the Targets window.

The extension includes the following items:

Item	Name
Body template	PAPALCCB.pls
Specification template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	target_extn

Table 9 – 17 Allocation Target Extension (Page 1 of 1)

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_alloc_rule_id	IN	Number	Identifies the allocation rule
x_target_proj_task_tbl	OUT	alloc_target_tabtype	Number the index sequentially from 1. Otherwise the process will fail.
x_error_message	OUT	VARCHAR2(30)	Error message text
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error

Table 9 – 18 Allocation Target Extension Parameters (Page 1 of 1)

Additional Parameter Information

The datatype *alloc_target_tabtype* contains the following parameters:

Parameter	Type	Description
project_id	Number	(Required) Identifies the target project. If cross–charging is enabled, target projects and source projects can be in different operating units.
task_id	Number	Identifies the target task (task must be chargeable)
percent	Number	The percentage of the pool amount allocated to this target. Express the value in numbers between 0 and 100 (for example, 45% is 45, not .45). NVL (percent,0). See Note on the Percent Parameter: page 9 – 45.
exclude_flag	Varchar2(1)	(Default is N) If Y, exclude the project and task from the target project and tasks

Table 9 – 19 Additional Parameters: Allocation Target Extension (Page 1 of 1)

Note on the Percent Parameter

If you want to use target percentages in a rule, specify the percentages either in the Targets window or within the extension, but not both. The Generate Allocation Transactions process ignores any target percentages in the rule if all of the following are true:

- The basis method for the allocation rule is *Target % and Spread Evenly* or *Target % and Prorate*
- The Targets window for the rule includes target lines.
- The client extension returns target percentages.

Validation

The Generate Allocation Transactions process:

- Validates project_id against view pa_alloc_target_proj_v
- Validates task_id against view pa_alloc_tgt_tasks_v
- Verifies that the task belongs to the target project

If the validation fails, the Generate Allocation Transactions process populates the message ”The client extension returned an invalid project or task.”

Allocation Offset Tasks Extension

This extension defines offset tasks. Oracle Projects calls this extension when Use Client Extension for Task is selected in the Offsets window. Use the Allocation Offset Tasks extension when you want to offset some source tasks but not others.

Description

The extension includes the following items:

Item	Name
Body template	PAPALCCB.pls
Specification template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	offset_task_extn

Table 9 – 20 Allocation Offset Tasks Extension (Page 1 of 1)

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_alloc_rule_id	IN	Number	(Required) Identifies the allocation rule
p_offset_project_id	IN	Number	(Required) Identifies the offset project
x_offset_task_id	OUT	Number	(Required) Identifies the offset task
x_error_message	OUT	VARCHAR2(30)	Error message text
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error

Table 9 – 21 Allocation Offset Tasks Extension Parameters (Page 1 of 1)

Validation

The Generate Allocation Transactions process:

- Validates task_id against pa_alloc_tgt_tasks_v
- Verifies that the returned tasks belong to the offset project that was provided as the input parameter

If the validation fails, the Generate Allocation Transactions process populates the message ”The client extension returned an invalid project or task.”

Allocation Offset Projects and Tasks Extension

This extension defines offset projects and tasks. Oracle Projects calls this extension when Use Client Extension for Project and Task is selected in the Offsets window.

Use this extension to specify more or different projects and tasks than are defined in the Sources window.

Description

For each allocation rule_id, the client populates the global session variable x_offset_proj_task_tbl of data type table alloc_offset_tabtype. The allocation run process reads the table to get the offset project, task, and offset amount for the allocation run. The sum of offset amounts assigned to each offset project and task equals the total offset amount (p_offset_amount).

The extension includes the following items:

Item	Name
Body Template	PAPALCCB.pls
Specification Template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	offset_extn

Table 9 – 22 Allocation Offset Projects and Tasks Extension

Parameters

Parameter	Usage	Type	Description
p_alloc_rule_id	IN	Number	(Required) Identifies the allocation rule
p_offset_amount	IN	Number	(Required) The pool amount to be offset
x_offset_proj_task_tbl	OUT	alloc_offset_tabtype	Number the index sequentially from 1. Otherwise the process will fail.
x_error_message	OUT	VARCHAR2(30)	Error message text
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error

Table 9 – 23 Allocation Offset Projects and Tasks Extension Parameters (Page 1 of 1)

Additional Parameter Information

The datatype **alloc_offset_tabtype** contains the following parameters:

Parameter	Type	Description
project_id	Number	(Required) Identifies the offset project. The offset project and the allocation rule must be from the same operating unit. The offset project must allow new transactions.
task_id	Number	(Required) Identifies the offset task (must be chargeable)
offset_amount	Number	(Required) The amount allocated to this project and task (Nvl(offset_amount,0)

Table 9 – 24 (Page 1 of 1)

Validation

The Generate Allocation Transactions process:

- Validates the project_id against the single organization view pa_projects

- Verifies that the project allows new transactions (that is, pa_project_utils.check_prj_stus_action_allowed (project_status_code,'NEW_TXNS')='Y' and template_flag !='Y')
- Validates task_id against pa_alloc_tgt_tasks_v
- Verifies that the task belongs to the offset project
- Validates the sum of the offset amount from client extension against p_offset_amount

If the validation fails, the Generate Allocation Transactions process populates one of these messages:

- 'The client extension returned an invalid project or task."
- 'The sum of offset amounts returned from the offset client extension does not equal the total offset amount passed to the client extension."

Allocation Basis Extension

Oracle Projects calls this extension when Use Client Extension Basis is selected in the Allocation Rule window. During the allocation run, the system calls the procedure to get the basis amount for each target project and task.

Use the Basis extension when you want to use amounts other than target costs to calculate the basis rate for target projects and tasks. For example, you may want to base the calculation on the number of people in a department, or the amount of floor space.

Description

The extension includes the following items:

Item	Name
Body template	PAPALCCB.pls
Specification template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	basis_extn

Table 9 – 25 Allocation Basis Extension (Page 1 of 1)

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_alloc_rule_id	IN	Number	(Required) Identifies the allocation rule
p_project_id	IN	Number	(Required) Identifies the offset project
p_task_id	IN	Number	(Required) Identifies the offset task
x_basis_amount	OUT	Number	(Required) The percentage of the pool amount allocated to this offset. NVL(x_basis_amount,0). Individual amounts can be negative or 0, but the sum of the basis amounts cannot equal zero.
x_error_message	OUT	VARCHAR2(30)	Error message text
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error

Table 9 – 26 Allocation Basis Extension Parameters (Page 1 of 1)

Validation

The Generate Allocation Transactions process validates the sum of basis amount returned from the client extension.

If the validation fails, the Generate Allocation Transactions process populates the message ”The total basis amount cannot be 0. No allocation can be performed.”

Allocation Descriptive Flexfields Extension

Use the Allocation Descriptive Flexfields extension to define descriptive flexfields to be used when defining allocation rules. The descriptive flexfields you define are used in creating allocation and offset transactions.

Description

Oracle Projects calls this extension before creating each transaction. If the extension provides descriptive flexfield values, the system uses the values when creating the transactions.

The extension includes the following items:

Item	Name
Body template	PAPALCCB.pls
Specification template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	txn_dff_extn

Table 9 – 27 Allocation Descriptive Flexfields Extension (Page 1 of 1)

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_rule_id	IN	Number	Identifies the allocation rule
p_run_id	IN	Number	The allocation run ID
p_txn_type	IN	VARCHAR2(1)	T=Target transaction O=Offset transaction
p_project_id	IN	Number	Identifies the offset project
p_task_id	IN	Number	Identifies the offset task
p_expnd_org	IN	VARCHAR2(30)	The expenditure organization associated with the transaction
p_expnd_type_class	IN	Number	The expenditure type class associated with the transaction
p_expnd_type	IN	VARCHAR2(30	The expenditure type
x_attribute_category	OUT	VARCHAR2(30	Descriptive flexfield context field
x_attribute1–10	OUT	VARCHAR2(150)	Descriptive flexfield segments

Table 9 – 28 Descriptive Flexfields Extension Parameters (Page 1 of 2)

Parameter	Usage	Type	Description
x_error_message	OUT	VARCHAR2(30)	Error message text
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error

Table 9 – 28 Descriptive Flexfields Extension Parameters (Page 2 of 2)

Allocation Dependencies Extension

Use the Allocation Dependencies extension to verify compliance with the business rules of your choice. For example, you could verify that certain projects or tasks are never included in a source pool, or that the previous allocation run used a particular rule.

Description

Oracle Projects calls this extension before processing any allocation rule. If the status code is zero (that is, if the dependencies specified in the extension are met) then the process creates an allocation run. If the status code is other than zero, the system prints the message provided by the x_message parameter.

The extension includes the following items:

Item	Name
Body template	PAPALCCB.pls
Specification template	PAPALCCS.pls
Package	pa_client_extn_alloc
Procedure	check_dependency

Table 9 – 29 Allocation Dependencies Extension (Page 1 of 1)

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_alloc_rule_id	IN	Number	(Required) Identifies the allocation rule
x_status	OUT	Number	(Required) Indicates if an error occurred: =0 Successful validation <0 Oracle error; message is written to a log file >0 Application error
x_error_message	OUT	VARCHAR2(30)	Error message text

Table 9 – 30 Allocation Dependencies Extension Parameters (Page 1 of 1)

Asset Allocation Basis Extension

This extension enables you to define your own allocation bases for allocating unassigned and common costs across multiple project assets.

See Also

Implementing Client Extensions: page 7 – 5

Allocating Asset Costs, *Oracle Project Costing User Guide*

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PACCXAAB.pls
Specification template	PACCXAAS.pls
Package	pa_client_extn_asset_alloc
Procedure	asset_alloc_basis

Table 9 – 31

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Business Rules

This extension is called by the PA_ASSET_ALLOCATION_PVT.ALLOCATE_UNASSIGNED procedure. It is called once for every *unassigned* asset line where the project (or batch) has an Asset Allocation Method equal to CE (Client Extension Basis). It enables you to determine the Total Basis Amount and the Asset Basis Amount for each asset in the array.

The p_asset_basis_table is passed to the Client Extension procedure. It is a table indexed by Binary Integer with three columns:

- PROJECT_ASSET_ID NUMBER;
- ASSET_BASIS_AMOUNT NUMBER
- TOTAL_BASIS_AMOUNT NUMBER

The table will already be populated with values for Project Asset ID, which correspond to the assets associated with the current *unassigned* asset line via Grouping Levels and Asset Assignments. The basis amount columns will contain zeros, which are then replace with values determined by this extension. The Total Basis Amount should be identical for each row in the table. You create the logic for determining the basis amounts for each asset.

Checks are performed on each project asset to verify that:

- Each project asset ID is valid for the project
- The Date Placed in Service is specified
- The Capital Hold flag is set to N, indicating that the asset is eligible for new asset line generation
- The Project Asset Type is AS–BUILT for capital asset lines (line type = C)
- The Project Asset Type is RETIREMENT_ADJUSTMENT for retirement cost asset lines (line type = R)

If you modify or add to assets in the P_ASSET_BASIS table, you must ensure that above conditions are true for each asset.

The following additional validations are also performed:

- The Total Basis Amount is not equal to zero (to avoid division by zero)
- Each Asset Basis Amount is not null and is not negative
- Each project asset in the array refers to the same Total Basis Amount
- The Asset Basis Amounts sum up to the Total Basis Amount

The Total Basis Amount is the sum of all Asset Basis Amounts in the table, and it is stored on each row. The asset allocation uses the Asset Basis Amount / Total Basis Amount for each project asset to prorate the amount of each *unassigned* asset line.

Asset Allocation Basis Procedure

The procedure name is: **asset_alloc_basis**

Use this procedure to define your own allocation bases for allocating unassigned and common costs across multiple project assets. Oracle Projects calls this procedure to allocate costs for projects that specify an asset cost allocation method of *Client Extension* in the Capital Information window.

Parameters

The following table lists the parameters that are used by the Asset Allocation Basis procedure.

Parameter	Usage	Type	Description
p_project_asset_line_id	IN	NUMBER	Project asset line identifier
p_project_id	IN	NUMBER	Project identifier
p_asset_basis_table	IN OUT	TABLE	Array of assets associated with the UNASSIGNED asset line via the grouping level
x_return_status	OUT	VARCHAR2	Mandatory OUT parameter indicating the return status of the API. Valid values are: S for Success, E for Error, and U for Unexpected Error.
x_msg_count	OUT	NUMBER	Indicates the error message count
x_msg_data	OUT	VARCHAR2	Indicates the error message text if there is only one error

Table 9 – 32 Asset Allocation Basis Procedure Parameters (Page 1 of 1)

Asset Assignment Extension

If the Generate Asset Lines process is unable to assign an asset to a task, the system marks the line as UNASSIGNED in the Asset Name column of the report.

Oracle Projects calls the Asset Assignment extension:

- For all unassigned assets. You can modify the extension to designate the assets for specific tasks (asset lines) and thus avoid the UNASSIGNED designation, or you can assign an asset to the line manually.
- If the Override Asset Assignment check box is selected on the Project Types window (Capitalization tab). You can modify the extension to override the asset assigned to specified tasks.

The asset you designate must:

- Be placed in service before the date identified by the In Service Through date in the Generate Asset Lines process
- Belong to the same project as the identified task

The extension includes an example that you can copy and modify.

Description

The extension includes the following items:

Item	Name
Body template	PAPGALCB.pls
Specification template	PAPGALCS.pls
Package	pa_client_extn_gen_asset_lines
Procedure	client_asset_assignment

Table 9 – 33 Asset Assignment Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifies the project
p_task_id	IN	NUMBER	Identifies the task
p_expnd_id	IN	NUMBER	Identifies the expenditure
p_expnd_item_id	IN	NUMBER	Identifies the expenditure item
p_expnd_type	IN	VARCHAR2	Expenditure type
p_expnd_category	IN	VARCHAR2	Expenditure category
p_expnd_type_class	IN	VARCHAR2	Expenditure type class
p_non_labor_org_id	IN	NUMBER	Identifies the organization for non–labor tasks
p_non_labor_resource	IN	VARCHAR2	Identifies the organization for non–labor resources
p_invoice_id	IN	NUMBER	Identifies the invoice
p_inv_dist_line_number	IN	NUMBER	Identifies the invoice distribution line
p_vendor_id	IN	VARCHAR2	Identifies the supplier
p_employee_id	IN	VARCHAR2	Identifies the employee
p_attribute1–10	IN	VARCHAR2	Descriptive flexfield segments
p_attribute_category	IN	VARCHAR2	Descriptive flexfield category
p_in_service_thorough _date	IN	DATE	Date through which the asset is in service
x_asset_id	IN OUT	NUMBER	Identifies the asset

Table 9 – 34 Asset Assignments Extension Parameters (Page 1 of 1)

Validation

You can validate the asset identifier (asset_id) in the client extension body to avoid exceptions during the PRC: Generate Asset Lines process.

If you do not do the validation in the client extension body, the system validates the asset identifier after the extension returns it. The

Generate Asset Lines exception report lists the lines that fail validation.

See Also

Generate Asset Lines, *Oracle Projects Fundamentals*

Asset Lines Processing Extension

This extension is called by the *PRC: Generate Asset Lines* process (for a Single Project or a Range of Projects) for each project for which asset lines are generated. You can use this extension to create project assets (capital assets and retirement adjustment assets) and asset assignments automatically prior to the creation of asset lines, based on transaction data (such as inventory issues or supplier invoices) entered for the project.

See Also

Implementing Client Extensions: page 7 – 5

Generating Summary Asset Lines, *Oracle Project Costing User Guide*

Generate Asset Lines, *Oracle Projects Fundamentals*

Location and Package Name

The extension includes the following items:

Item	Name
Body template	PACCXACB.pls
Specification template	PACCXACS.pls
Package	pa_client_extn_asset_creation
Procedure	create_project_assets

Table 9 – 35

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Asset Lines Processing Procedure

The procedure name is: **create_project_assets**

When you submit the *PRC: Generate Asset Lines* process (for a Single Project or a Range of Projects), Oracle Projects calls this procedure for each project prior to creating asset lines. The intended use of this extension is to automatically create project assets (capital assets and retirement adjustment assets) and asset assignments prior to the creation of asset lines, based on transaction data (such as inventory issues or supplier invoices) entered for the project.

Parameters

The following table lists the parameters that are used by the Asset Lines Processing procedure.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	project identifier
p_asset_through_date	IN	DATE	Runtime parameter for the PRC: Generate Asset Lines process
pa_date_through	IN	DATE	Runtime parameter for the PRC: Generate Asset Lines process
p_capital_event_id	IN	DATE	Runtime parameter for the PRC: Generate Asset Lines process
x_return_status	OUT	VARCHAR2	Mandatory OUT parameter that indicates the return status of the API. Valid values are: S–Success, E–Error, and U–Unexpected error.
x_msg_data	OUT	VARCHAR2	Indicates the error message text

Table 9 – 36 Asset Lines Processing Procedure Parameters (Page 1 of 1)

Capital Event Processing Extension

This extension is called by the *PRC: Create Periodic Capital Event* process for each project for which a capital event is created. You can use this extension to create project assets (capital assets and retirement adjustment assets) and asset assignments automatically prior to the creation of capital events, based on transaction data (such as inventory issues or supplier invoices) entered for the project.

See Also

Implementing Client Extensions: page 7 – 5

Creating Capital Events, *Oracle Project Costing User Guide*

Create Periodic Capital Events, *Oracle Projects Fundamentals*

Location and Package Name

The extension includes the following items:

Item	Name
Body template	PACCXCBB.pls
Specification template	PACCXCBS.pls
Package	pa_client_extn_pre_cap_event
Procedure	pre_capital_event

Table 9 – 37

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Capital Event Processing Procedure

The procedure name is: **pre_capital_event**

When you submit the *PRC: Create Periodic Capital Event* process, Oracle Projects calls this procedure for each project prior to creating a capital event. The intended use of this extension is to automatically create project assets (capital assets and retirement adjustment assets) and asset assignments prior to the creation of capital events, based on transaction data (such as inventory issues or supplier invoices) entered for the project.

Parameters

The following table lists the parameters that are used by the Capital Event Processing procedure.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project
p_event_period_name	IN	VARCHAR2(15)	Runtime parameter for the PRC: Create Periodic Capital Events process
p_asset_date_through	IN	DATE	Runtime parameter for the PRC: Create Periodic Capital Events process
p_ei_date_through	IN	DATE	Runtime parameter for the PRC: Create Periodic Capital Events process
x_return_status	OUT	VARCHAR2	Mandatory OUT parameter that indicates the return status of the API. Valid values are: S–Success, E–Error, and U–Unexpected error.
x_msg_data	OUT	VARCHAR2	Indicates the error message text

Table 9 – 38 Capital Event Processing Procedure Parameters (Page 1 of 1)

Capitalized Interest Extension

The capitalized interest client extension enables you to customize the capitalized interest calculation process.

See Also

Implementing Client Extensions: page 7 – 5

Capitalizing Interest, *Oracle Project Costing User Guide*

Capitalized Interest, *Oracle Projects Implementation Guide*

Location and Package Name

The extension includes the following items:

Item	Name
Body template	PACINTXB.pls
Specification template	PACINTXS.pls
Package	pa_client_extn_cap_int

Table 9 – 39

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Procedures

The following procedures are provided in the capitlized interest client extension.

Target Task Override Procedure

The procedure name is get_target_task

The Target Task Override procedure enables you to redirect capitalized interest transactions to specific tasks.

Parameters

The following table lists the parameters that are used by the Target Task Override procedure.

Parameter	Usage	Type	Description
P_SOURCE_TASK_ID	IN	NUMBER	Task identifier
P_SOURCE_TASK_NUM	IN	VARCHAR2	Task number
P_RATE_NAME	IN	VARCHAR2	Rate name
X_TARGET_TASK_ID	OUT	NUMBER	Target task identifier
X_TARGET_TASK_NUM	OUT	VARCHAR2	Target task number
X_RETURN_STATUS	OUT	VARCHAR2	Mandatory OUT parameter indicating the return status of the API. Valid values are: S for Success, E for Error, and U for Unexpected Error.
X_ERROR_MSG_COUNT	OUT	NUMBER	Indicates the error message count
X_ERROR_MSG_CODE	OUT	VARCHAR2	Indicates the error message code

Table 9 – 40 Target Task Override Procedure Parameters (Page 1 of 1)

Expenditure Organization Procedure

The procedure name is: **expenditure_org**

The Expenditure Organization procedure enables you to specify organizations other than the source project owning organization or source task owning organization as the expenditure organization for generated transactions.

Parameters

The following table lists the parameters that are used by the Expenditure Organization procedure.

Parameter	Usage	Type	Description
P_EXPENDITURE_ITEM_ID	IN	NUMBER	Expenditure item identifier
P_LINE_NUM	IN	NUMBER	CDL line number
P_RATE_NAME	IN	VARCHAR2	Rate name

Table 9 – 41 Expenditure Organization Procedure Parameters (Page 1 of 1)

Interest Rate Multiplier Override Procedure

The procedure name is rate_multiplier.

The Interest Rate Multiplier Override procedure enables you to define multiple interest rate multipliers based on the rate name and task owning organization.

Parameters

The following table lists parameters that are used by the Interest Rate Multiplier Override procedure.

Parameter	Usage	Type	Description
P_EXPENDITURE_ITEM_ID	IN	NUMBER	Expenditure item identifier
P_LINE_NUM	IN	NUMBER	CDL line number
P_RATE_NAME	IN	VARCHAR2	Rate name

Table 9 – 42 Interest Rate Multiplier Override Procedure Parameters (Page 1 of 1)

Interest Override Procedure

The procedure name is calculate_cap_interest.

The Interest Override procedure enables you to define your own calculations for capitalized interest.

Parameters

The following table lists the parameters that are used by the Interest Override procedure.

Parameter	Usage	Type	Description
P_GL_PERIOD	IN	VARCHAR2	GL period name
P_RATE_NAME	IN	VARCHAR2	Rate name
P_CURR_PERIOD_MULT	IN	NUMBER	Current period multiplier
P_PERIOD_MULT	IN	NUMBER	Period multiplier
P_PROJECT_ID	IN	NUMBER	Project identifier
P_SOURCE_TASK_ID	IN	NUMBER	Task identifier
P_TARGET_TASK_ID	IN	NUMBER	Target task identifier
P_EXP_ORG_ID	IN	NUMBER	Expenditure organization identifier
P_EXP_ITEM_DATE	IN	DATE	Expenditure item date
P_PRIOR_PERIOD_AMT	IN	NUMBER	Prior basis amount for capitalized interest calculation
P_CURR_PERIOD_AMT	IN	NUMBER	Current basis amount for capitalized interest calculation
P_GROUPING_METHOD	IN	VARCHAR2	Grouping method
P_RATE_MULT_	IN	NUMBER	Rate multiplier

Table 9 – 43 Interest Override Procedure Parameters (Page 1 of 2)

Parameter	Usage	Type	Description
X_CAP_INT_AMT	IN OUT	NUMBER	Capitalized interest amount
X_RETURN_STATUS	OUT	VARCHAR2	Mandatory OUT parameter indicating the return status of the API. Valid values are: S for Success, E for Error, and U for Unexpected Error.
X_ERROR_MSG_COUNT	OUT	NUMBER	Indicates the error message count
X_ERROR_MSG_CODE	OUT	VARCHAR2	Indicates the error message code

Table 9 – 43 Interest Override Procedure Parameters (Page 2 of 2)

Interest Threshold Procedure

The procedure name is: **check_thresholds**

The Interest Threshold procedure enables you to define duration and amount thresholds at levels lower than the operating unit.

Parameters

The following table lists the parameters that are used by the Interest Threshold procedure.

Parameter	Usage	Type	Description
P_PROJECT_ID	IN	NUMBER	Project identifier
P_TASK_ID	IN	NUMBER	Task identifier
P_RATE_NAME	IN	VARCHAR2	Rate name

Table 9 – 44 Interest Threshold Procedure Parameters (Page 1 of 2)

Parameter	Usage	Type	Description
P_START_DATE	IN	DATE	Start date of the GL period
P_END_DATE	IN	DATE	End date of the GL period
P_THRESHOLD_AMT_TYPE	IN	VARCHAR2	Threshold amount type
P_BUDGET_TYPE	IN	VARCHAR2	Budget type
P_FIN_PLAN_TYPE_ID	IN	NUMBER	Financial plan type identifier
P_INTEREST_CALC_METHOD	IN	VARCHAR2	Interest calculation method
P_CIP_COST_TYPE	IN	VARCHAR2	CIP cost type
X_DURATION_THRESHOLD	IN OUT	NUMBER	Duration threshold
X_AMT_THRESHOLD	IN OUT	NUMBER	Amount threshold
X_RETURN_STATUS	OUT	VARCHAR2	Mandatory OUT parameter indicating the return status of the API. Valid values are: S for Success, E for Error, and U for Unexpected Error.
X_ERROR_MSG_COUNT	OUT	NUMBER	Indicates the error message count
X_ERROR_MSG_CODE	OUT	VARCHAR2	Indicates the error message code

Table 9 – 44 Interest Threshold Procedure Parameters (Page 2 of 2)

Grouping Method Procedure

The procedure name is grouping_method.

The Grouping Method procedure enables you to specify grouping criteria.

Parameters

The following table lists the parameters that are used by the Grouping Method procedure.

Parameter	Usage	Type	Description
P_GL_PERIOD	IN	VARCHAR2	GL period name
P_PROJECT_ID	IN	NUMBER	Project identifier
P_SOURCE_TASK_ID	IN	NUMBER	Task identifier
P_EXPENDITURE_ITEM_ID	IN	NUMBER	Expenditure item identifier
P_LINE_NUM	IN	NUMBER	CDL line number
P_EXPENDITURE_ID	IN	NUMBER	Expenditure identifier
P_EXPENDITURE_TYPE	IN	VARCHAR2	Expenditure type
P_EXPENDITURE_CATEGORY	IN	VARCHAR2	Expenditure category
P_TRANSACTION_SOURCE	IN	VARCHAR2	Transaction source
P_RATE_NAME	IN	VARCHAR2	Rate name
P_ATTRIBUTE1 through P_ATTRIBUTE10	IN	VARCHAR2	Attribute 1 through Attribute 10
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	Attribute category

Table 9 – 45 Grouping Method Procedure Parameters (Page 1 of 1)

Get Transaction Attributes Procedure

The procedure name is: **get_txn_attribute**

The Get Transaction Attributes procedure enables you to control how the transaction attribute columns are populated.

Parameters

The following table lists the parameters that are used by the Get Transaction Attributes procedure.

Parameter	Usage	Type	Description
P_PROJECT_ID	IN	NUMBER	Project identifier
P_SOURCE_TASK_ID	IN	NUMBER	Source task identifier
P_TARGET_TASK_ID	IN	NUMBER	Target task identifier
P_RATE_NAME	IN	VARCHAR2	Rate name
P_GROUPING_METHOD	IN	VARCHAR2	Grouping method
X_ATTRIBUTE1 through X_ATTRIBUTE10	OUT	VARCHAR2	Attribute 1 through Attribute 10
X_ATTRIBUTE_CATEGORY	OUT	VARCHAR2	Attribute category
X_RETURN_STATUS	OUT	VARCHAR2	Mandatory OUT parameter indicating the return status of the API. Valid values are: S for Success, E for Error, and U for Unexpected Error.
X_ERROR_MSG_COUNT	OUT	NUMBER	Indicates the error message count
X_ERROR_MSG_CODE	OUT	VARCHAR2	Indicates the error message code

Table 9 – 46 Get Transaction Attributes Procedure Parameters (Page 1 of 1)

CIP Grouping Extension

Use the CIP (Construction–In–Process) Grouping extension to define a unique method that your company uses to specify how expenditure lines are grouped to form asset lines.

Oracle Projects predefines five CIP Grouping Methods. If these methods do not meet your company's business needs, use this client extension to create your own CIP Grouping Method. Once the extension has been created, you can assign the grouping method to individual projects by selecting the "Group by Client Extension" grouping method in the Capitalization tab of the Project Types window.

Oracle Projects calls the CIP Grouping extension during the Generate Asset Lines process.

Description

The extension is identified by the following items:

Item	Name
Specification template	PAXGCES.pls
Body template	PAXGCEB.pls
Package	pa_client_exten_cip_grouping
Function	client_grouping_method

Table 9 – 47 CIP Grouping Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Client_Grouping_Method Function

The client_grouping_method function uses the following parameters:

Parameter	Usage	Type	Description
p_proj_id	IN	NUMBER	Identifies the project.
p_task_id	IN	NUMBER	Identifies the task.

Parameter	Usage	Type	Description
p_expnd_id	IN	NUMBER	Identifies the expenditure.
p_expnd_item_id	IN	NUMBER	Identifies the expenditure item.
p_expnd_type	IN	VARCHAR2	Identifies the expenditure type.
p_expnd_category	IN	VARCHAR2	Identifies the expenditure category.
p_attribute1 – p_attribute10	IN	VARCHAR2	Identifies the descriptive flex fields.
p_attribute_category	IN	VARCHAR2	Identifies the descriptive flex category.
p_transaction_source	IN	VARCHAR2	Identifies the transaction source.

Table 9 – 48 client_grouping_method parameters (Page 2 of 2)

See Also

Project Types: Capitalization Information, *Oracle Projects Implementation Guide*

Creating a Capital Asset in Oracle Projects, *Oracle Project Costing User Guide*

Example of Using the Asset Line Grouping Extension

The body template, PAXGCEB.pls, includes a sample PL/SQL procedure for defining a CIP grouping method. The sample grouping method groups asset lines by material expenditures and non–material expenditures.

The sample procedure is shown below.

```
CREATE OR REPLACE
Package BODY PA_CLIENT_EXTEN_CIP_GROUPING
AS
FUNCTION CLIENT_GROUPING_METHOD(
 p_proj_id        IN    PA_PROJECTS_ALL.project_id%TYPE,
 p_task_id        IN PA_TASKS.task_id%TYPE,
 p_expnd_item_id  IN PA_EXPENDITURE_ITEMS_ALL.expenditure_item_id%TYPE,
```

```
 p_expnd_id       IN PA_EXPENDITURE_ITEMS_ALL.expenditure_id%TYPE,
 p_expnd_type     IN PA_EXPENDITURE_TYPES.expenditure_type%TYPE,
 p_expnd_category IN PA_EXPENDITURE_CATEGORIES.expenditure_category%TYPE,
 p_attribute1     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute2     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute3     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute4     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute5     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute6     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute7     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute8     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute9     IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute10    IN PA_EXPENDITURE_ITEMS_ALL.attribute1%TYPE,
 p_attribute_category IN PA_EXPENDITURE_ITEMS_ALL.attribute_category%TYPE,
 p_transaction_source IN PA_EXPENDITURE_ITEMS_ALL.transaction_source%TYPE)
 return VARCHAR2 IS
v_grouping_method      varchar2(2000);
v_material_flag        pa_expenditure_types.attribute10%TYPE;
BEGIN
  /*Assume CIP grouping method is by default made up of attribute 6 to attribute
    10 in the following order:8,9,10,6,7 */
    v_grouping_method := p_attribute8||p_attribute9||p_attribute10||
                         p_attribute6||p_attribute7;

  /* In addition, the grouping method may have either expenditure type or
     material flag appended to it */
  /* If you want to further classify the grouping method by material flag, do the
     following and comment the 'grouping by expenditure type' section*/
      Select attribute10 into v_material_flag
      From PA_EXPENDITURE_TYPES
      Where expenditure_type = p_expnd_type;
      if (v_material_flag is not null ) then
        v_grouping_method := v_grouping_method || v_material_flag;
      end if;

   /* If you want to further classify the grouping method by Expenditure Type,
      uncomment the following and comment 'grouping by material
      flag' section */
     -- v_grouping_method := v_grouping_method || p_expnd_type

   /* If grouping method is null then return ALL*/
      IF v_grouping_method is null then
         v_grouping_method := 'ALL';
      end if;
```

```
    return v_grouping_method;
  ----v_grouping_method stores the grouping method to be returned by the function

    EXCEPTION
    WHEN OTHERS THEN
      null;
    END;
 END PA_CLIENT_EXTEN_CIP_GROUPING;
/
commit;
exit;
```

CIP Account Override Extension

You can use this extension to override the CIP account associated with an asset line and to specify a different account for posting CIP clearing amounts. This enables you to:

- Use accounts for clearing CIP amounts that are different from the accounts you use to account for CIP expenditures
- Preserve the original CIP cost account details

See Also

Implementing Client Extensions: page 7 – 5

Creating and Preparing Asset Lines for Oracle Assets, *Oracle Project Costing User Guide*

Location and Package Name

The extension includes the following items:

Item	Name
Body template	PACCXCOB.pls
Specification template	PACCXCOS.pls
Package	pa_client_extn_cip_acct_ovr
Procedure	cip_acct_override

Table 9 – 49

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

CIP Account Override Procedure

The procedure name is: **cip_acct_override**

Use this procedure to override the CIP account associated with an asset line to specify a different account for posting CIP clearing amounts. Oracle Projects calls this procedure when you submit the PRC: Generate Asset Lines process.

Parameters

The following table lists the parameters that are used by the CIP Account Override procedure.

Parameter	Usage	Type	Description
p_cdl_cip_ccid	IN	NUMBER	CIP account defined on the cost distribution line
p_expenditure_item_id	IN	NUMBER	Cost distribution line expenditure item ID
p_cdl_line_number	IN	NUMBER	Cost distribution line number
[return]	OUT	NUMBER	The function returns the override CCID value.

Table 9 – 50 CIP Account Override Procedure Parameters (Page 1 of 1)

Depreciation Account Override Extension

This extension enables you to specify logic for deriving the depreciation expense account assigned to a project asset.

See Also

Implementing Client Extensions: page 7 – 5

Defining and Processing Assets, *Oracle Project Costing User Guide*

Interface Assets, *Oracle Projects Fundamentals.*

Business Rules

This extension is called by the PRC: Interface Assets process.

Before the process validates that the Depreciation Expense CCID is populated, it calls this extension if the Book Type Code and Asset Category are NOT NULL. If a valid value is returned, the value is updated on the Project Asset.

The extension calls a procedure that checks to see that the new value returned is a valid CCID for the current Chart of Accounts.

Location and Package Name

The extension includes the following items:

Item	Name
Body template	PACCXDEB.pls
Specification template	PACCXDES.pls
Package	pa_client_extn_deprn_exp_ovr
Procedure	deprn_exp_acct_override

Table 9 – 51

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Depreciation Account Override Procedure

The procedure name is: **deprn_exp_acct_override**

Use this procedure to define your own logic for deriving the depreciation expense account assigned to a project asset. Oracle Projects calls this procedure during update of the Assets and Asset Details windows, and during validation of asset information when you submit the PRC: Interface Assets process.

Parameters

The following table lists the parameters that are used by the Depreciation Account Override procedure.

Parameter	Usage	Type	Description
p_project_asset_id	IN	NUMBER	Project asset identifier
p_book_type_code	IN	VARCHAR2	Asset book identifier
p_asset_category_id	IN	NUMBER	Asset category identifier
p_date_placed_in_service	IN	DATE	Date placed in service
p_deprn_expense_acct_ccid	IN	NUMBER	Current depreciation expense account identifier

Table 9 – 52 Depreciation Account Override Procedure Parameters (Page 1 of 1)

Cross–Charge Client Extensions

You can implement your business rules for various aspects of cross charge feature by using the following client extensions:

Provider and Receiver Organizations Override Extension: page 9 – 81

Cross Charge Processing Method Override Extension: page 9 – 83

Transfer Price Determination Extension: page 9 – 86

Transfer Price Override Extension: page 9 – 89

Transfer Price Currency Conversion Override Extension: page 9 – 92

Internal Payables Invoice Attributes Override Extension: page 9 – 94

See Also

Cost Accrual Identification Extension: page 10 – 52

Provider and Receiver Organizations Override Extension

You can use this client extension to enforce cross–charge rules at a higher level in the organization hierarchy than the level at which you assign resources and projects. Doing so provides a single place for you to enforce and maintain your business rules in all organizations in your enterprise.

The system identifies cross–charged transactions based on the provider and receiver organizations for the transaction. It derives default values for these organizations as follows:

- Provider organization: The expenditure organization or non–labor resource organization for usage transactions
- Receiver organization: The organization that owns the task to which the transaction is charged

To override the cross–charge identification, code this extension to use a higher level in the organization hierarchy to derive the appropriate provider and receiver organizations and then determine if a transaction is to be a cross–charged transaction.

When you run the cost distribution processes or use the Expenditure Items window to adjust cross–charged transactions, the system first identifies the default provider and receiver organizations for the transaction and then calls the extension.

Description

The extension is identified by the following items:

Item	Name
Body template	PACCIXTB.pls
Specification template	PAACCIXTS.pls
Package	PA_CC_IDENT_CLIENT_EXTN
Procedure	override_prvdr_recvr

Table 9 – 53 Provider and Receiver Organizations Override Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Parameters

The extension uses the following parameters:

Parameter	Type	Usage	Description
p_PrvdrOrganizationId	NUMBER	IN	Provider organization identifier (the expenditure organization is the default value)
p_PrvdrOrgId	NUMBER	IN	Provider operating unit identifier
p_RecvrOrganizationId	NUMBER	IN	Receiver organization identifier (the task organization is the default value).
p_RecvrOrgId	NUMBER	IN	Receiver operating unit identifier
p_TransId	NUMBER	IN	Expenditure item identifier
p_SysLink	NUMBER	IN	Expenditure type class of the Transaction
x_PrvdrOrganizationId	NUMBER	OUT	Return provider organization. Returns the input value by default
x_RecvrOrganizationId	NUMBER	OUT	Return receiver organization. Returns the input value by default
x_ErrorStage	VARCHAR2	OUT	Error message text
X_Status	NUMBER	OUT	Status indicating whether an error occurred. Valid values are: =0 Success <0 Oracle Error >0 Application Error

Table 9 – 54 Parameters for Provider and Receiver Organizations Override Extension

Validation

The system verifies the returned values to ensure that they are valid organizations within the business group.

Cross–Charge Processing Method Override Extension

You may have some custom business rules that help you identify how you want to process cross–charged transactions. You can use this extension to:

- Exclude certain cross–charged transactions from cross–charge processing
- Change the cross–charge method (for example, from Intercompany Billing to Borrowed and Lent accounting)

When you run a cost distribution process or use the Expenditure Items window to adjust cross–charged transactions, the system does the following:

1. Identifies the transaction as a cross–charged transaction
2. Determines the cross–charge processing method (based on how you set up the cross–charge options)
3. Calls the extension so you can override the cross–charge processing method

Description

The extension is identified by the following items:

Item	Name
Body template	PACCIXTB.pls
Specification template	PACCIXTS.pls
Package	PA_CC_IDENT_CLIENT_EXTN
Procedure	override_cc_processing_method

Table 9 – 55 Cross–Charge Processing Method Override Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Prerequisites

The transaction must be a cross–charged transaction.

Parameters

The extension uses the following parameters:

Parameter	Type	Usage	Description
p_PrvdrOrganizationId	NUMBER	IN	Provider organization identifier
p_RecvrOrganizationId	NUMBER	IN	Receiver organization identifier
p_PrvdrOrgId	NUMBER	IN	Provider operating unit identifier
p_RecvrOrgId	NUMBER	IN	Receiver operating unit identifier
p_PrvdrLeId	NUMBER	IN	Provider legal entity identifier
p_RecvrLeId	NUMBER	IN	Receiver legal entity identifier
p_ProjectId	NUMBER	IN	Project identifier
p_TaskId	NUMBER	IN	Task identifier
p_PersonId	NUMBER	IN	Identifier for employee incurring the transaction
p_SysLink	VARCHAR2	IN	Expenditure type class
p_TransDate	DATE	IN	Expenditure item date
p_TransSource	VARCHAR2	IN	External source of the transaction, if any
p_TransId	NUMBER	IN	Identifier of the transaction
p_CrossChargeType	VARCHAR2(2)	IN	Cross Charge Type determined for the transactions. Values are from the lookup CC_CROSS_CHARGE_TYPE
p_CrossChargeCode	VARCHAR2(1)	IN	Input value for Cross Charge Identification from the lookup CC_CROSS_CHARGE_CODE
x_OvrridCrossChargeCode	VARCHAR2(1)	OUT	Return value for Cross Charge Identification, which must be from the lookup CC_CROSS_ CHARGE_CODE. The default logic returns the input value.

Table 9 – 56 Parameters for Cross–Charge Processing Method Override Extension

Parameter	Type	Usage	Description
x_ErrorStage	VARCHAR2	OUT	Error message text
X_Status	NUMBER	OUT	Status indicating whether an error occurred. Valid values are: =0 Success <0 Oracle Error >0 Application Error

Table 9 – 56 Parameters for Cross–Charge Processing Method Override Extension

Validation

The system validates the value returned for the cross–charge code to ensure that it meets the following rules:

If the cross–charge type is...	The following processing methods are allowed:
Intra–Operating Unit (that is, within a single operating unit)	Borrowed and Lent None (no processing)
Inter–Operating Unit (that is, across operating units within a single legal entity)	Intercompany Billing Borrowed and Lent None (no processing)
Intercompany (that is, across legal entities)	Intercompany Billing None (no processing)

Table 9 – 57 Cross–Charge Types

Transfer Price Determination Extension

Although your transfer price setup determines the transfer price used for cross–charged transactions, you may want to enforce different business rules occasionally.

The extension **determine_transfer_price** specifies a transfer price for the transaction being processed. If this extension returns a valid value for the transfer price, Oracle Projects uses that value as the transfer price instead of computing the transfer price. The Distribute Borrowed and Lent Amounts and the Generate Intercompany Invoice processes call this extension, before calling the standard transfer price determination routine.

For another type of transfer price extension, see: Transfer Price Override Extension: page 9 – 89.

Description

This extension is identified by the following items:

Item	Name
Body template	PAPTPRCB.pls
Specification template	PAPTPRCS.pls
Package	PA_CC_TP_CLIENT_EXTN
Procedure	determine_transfer_price

Table 9 – 58 Transfer Price Determination Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Prerequisites

- Complete all the setup steps described in the Cross Charge – Intercompany Billing setup steps section in the Oracle Projects Implementation Checklist, *Oracle Projects Implementation Guide*.
- Run the cost distribution processes for new transactions or use the Expenditure Items window to perform cross–charge adjustments on existing transactions. Both processes identify cross–charge transactions.

- Run the processes PRC: Distribute Borrowed and Lent Amounts or PRC: Generate Intercompany Invoices to process transactions that are identified as cross charged and that require borrowed and lent or intercompany processing.

Parameters

The extension uses the following parameters:

Parameter	Type	Usage	Description
p_prvdr_org_id	NUMBER	IN	Provider operating unit identifier
p_project_id	NUMBER	IN	Project identifier
p_task_id	NUMBER	IN	Task identifier
p_recvr_org_id	NUMBER	IN	Operating unit identifier for the receiver project
p_prvdr_organization_id	NUMBER	IN	Unique identifier of the provider organization
p_recvr_organization_id	NUMBER	IN	Receiver organization identifier
p_transaction_type	VARCHAR2	IN	An identifier that distinguishes between actual and forecast transactions. This enables you to process forecast and actual transactions differently. The default value is ACTUAL. Forecast transactions have the transaction type FORECAST.
p_expenditure_item_id	NUMBER	IN	Expenditure item identifier
p_expnd_organization_id	NUMBER	IN	Expenditure organization identifier
p_expenditure_type_class	VARCHAR2	IN	The type class of the expenditure
p_expenditure_type	VARCHAR2	IN	Expenditure type
p_incurred_by_person_id	NUMBER	IN	Identifier of the person who incurred the expenditure
p_quantity	NUMBER	IN	Number of units of work performed
x_denom_transfer_price	NUMBER	OUT	Transfer price amount in transaction currency
x_denom_tp_curr_code	VARCHAR2	OUT	Transaction currency in which transfer price is calculated
x_tp_bill_rate	NUMBER	OUT	Bill rate applied to calculate the transfer price.

Table 9 – 59 Parameters for Transfer Price Determination Extension

Parameter	Type	Usage	Description
x_bill_markup_percentage	NUMBER	OUT	Percentage used in deriving the transfer price if the transfer price was based on a markup.
x_error_message	VARCHAR2	OUT	Error message text
X_Status	NUMBER	OUT	Status indicating whether an error occurred. Valid values are: =0 Success <0 Oracle Error >0 Application Error

Table 9 – 59 Parameters for Transfer Price Determination Extension

Validation

The system validates that you have provided a value for only one of the following output audit parameters:

- x_bill_rate
- x_bill_markup_percentage

Transfer Price Override Extension

Although your transfer price setup determines the transfer price used for cross–charged transactions, you may want to enforce different business rules occasionally. To do so, you can use the Transfer Price Override extension for a given transaction.

The extension (procedure) **override_transfer_price** overrides the transfer price for a transaction. After the Distribute Borrowed and Lent Amounts Process and Generate Intercompany Invoice Process compute the transfer price (as determined by the user setup in the Transfer Price Rules and Transfer Price Schedules windows), the processes call this extension.

For another type of transfer price extension, see: Transfer Price Determination Extension: page 9 – 86

Description

This extension is identified by the following items:

Item	Name
Body template	PAPTPRCB.pls
Specification template	PAPTPRCS.pls
Package	PA_CC_TP_CLIENT_EXTN
Procedure	override_transfer_price

Table 9 – 60 Transfer Price Override Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Prerequisites

- Complete all the setup steps described in the Cross Charge – Intercompany Billing setup steps section in the Oracle Projects Implementation Checklist, *Oracle Projects Implementation Guide*.
- Run the cost distribution processes for new transactions or use the Expenditure Items window to perform cross–charge adjustments on existing transactions. Both processes identify cross–charge transactions.

- Run the processes PRC: Distribute Borrowed and Lent Amounts or PRC: Generate Intercompany Invoices to process transactions that are identified as cross charged and that require borrowed and lent or intercompany processing.

Parameters

The extension uses the following parameters:

Parameter	Type	Usage	Description
p_prvdr_org_id	NUMBER	IN	Provider operating unit identifier
p_recvr_org_id	NUMBER	IN	Receiver organization unit identifier
p_prvdr_organization_id	NUMBER	IN	Provider organization identifier
p_recvr_organization_id	NUMBER	IN	Receiver organization identifier
p_project_id	NUMBER	IN	Project identifier
p_task_id	NUMBER	IN	Task identifier
p_transaction_type	VARCHAR2	IN	An identifier that distinguishes between actual and forecast transactions. This enables you to process forecast and actual transactions differently. The default value is ACTUAL. Forecast transactions have the following transaction type: FORECAST.
p_expenditure_item_id	NUMBER	IN	Expenditure item identifier
p_expnd_organization_id	NUMBER	IN	Expenditure organization identifier
p_expenditure_type_class	VARCHAR2	IN	The type class of the expenditure
p_expenditure_type	VARCHAR2	IN	Expenditure type
p_incurred_by_person_id	NUMBER	IN	Identifier of the person who incurred the expenditure
p_quantity	NUMBER	IN	Number of units of work performed
p_base_curr_code	VARCHAR2	IN	Transaction currency code of the base amount
p_base_amount	NUMBER	IN	Base amount used to derive the transfer price. It could be either a raw cost, burdened cost, or raw revenue in the transaction currencies.
p_denom_tp_curr_code	VARCHAR2	IN	Transaction currency code in which transfer price is calculated
p_denom_transfer_price	NUMBER	IN	Transfer price amount as calculated in the transaction currency

Table 9 – 61 Parameters for Transfer Price Override Extension

Parameter	Type	Usage	Description
x_denom_transfer_price	NUMBER	OUT	Transfer price as calculated in the transaction currency
x_denom_tp_curr_code	VARCHAR2	OUT	Transaction currency in which the transfer price is calculated
x_tp_bill_rate	NUMBER	OUT	Bill rate applied to calculate the transfer price.
x_bill_markup_percentage	NUMBER	OUT	Percentage used to derive the transfer price if the transfer price was based on a markup.
x_error_message	VARCHAR2	OUT	Error message text
X_Status	NUMBER	OUT	Status indicating whether an error occurred. Valid values are: =0 Success <0 Oracle Error >0 Application Error

Table 9 – 61 Parameters for Transfer Price Override Extension

Validation

The system validates that you have provided a value for only one of the following output audit parameters:

- x_bill_rate
- x_bill_markup_percentage

Transfer Price Currency Conversion Override Extension

Use this extension when you occasionally want to override the default attributes used to convert the transfer price from the transaction currency to the functional currency. The Distribute Borrowed and Lent Amounts and the Generate Intercompany Invoice Processes call the extension after the processes compute the transfer price. (The user setup in the Cross Charge tab in the Implementation Options window determines the default attributes used for the conversion.)

Description

The extension is identified by the following items:

Item	Name
Body template	PAPMCECB.pls
Specification template	PAPMCECS.pls
Package	PA_MULTI_CURR_CLIENT_EXTN
Procedure	override_curr_conv_attributes

Table 9 – 62 Transfer Price Currency Conversion Override Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Prerequisites

- Complete all the setup steps described in the Cross Charge – Intercompany Billing setup steps section in the Oracle Projects Implementation Checklist, *Oracle Projects Implementation Guide*.
- Run the cost distribution processes for new transactions or use the Expenditure Items window to perform cross–charge adjustments on existing transactions. Both processes identify cross–charge transactions.
- Run the processes PRC: Distribute Borrowed and Lent Amounts or PRC: Generate Intercompany Invoices to process transactions that are identified as cross charged and that require borrowed and lent or intercompany processing.

Parameters

The extension uses the following parameters:

Parameter	Type	Usage	Description
p_project_id	NUMBER	IN	Project identifier
p_task_id	NUMBER	IN	Task identifier
p_transaction_class	VARCHAR2	IN	The hard coded value ”Transfer Price”
p_expenditure_item_id	NUMBER	IN	Expenditure item identifier
p_expenditure_type_class	VARCHAR2	IN	The type class of the expenditure.
p_expenditure_type	VARCHAR2	IN	Expenditure type
p_expenditure_category	VARCHAR2	IN	Expenditure category
p_from_currency_code	VARCHAR2	IN	Currency to convert from
p_to_currency_code	VARCHAR2	IN	Currency to convert to
p_conversion_type	VARCHAR2	IN	Default exchange rate type to be used for conversion
p_conversion_date	DATE	IN	Default exchange Rate date to be used for conversion
x_rate_type	VARCHAR2	OUT	Override exchange rate type
X_rate_date	DATE	OUT	Override exchange rate date
X_exchange_rate	NUMBER	OUT	Override exchange rate to be used for rate type of ”USER” only.
x_error_message	VARCHAR2	OUT	Error message text
X_Status	NUMBER	OUT	Status indicating whether an error occurred. Valid values are: =0 Success <0 Oracle Error >0 Application Error

Table 9 – 63 Parameters for Transfer Price Override Extension

Validation

Oracle Projects validates that the values returned by the client extension meet all conversion requirements.

Internal Payables Invoice Attributes Override Extension

When using Intercompany or Inter–Project Billing, you must define organization controls using the Provider / Receiver Controls window. For each Provider and Receiver pair, you select the expenditure type and expenditure organization to use when creating the internal payables invoices. In order to further classify cost based on additional transaction information, you can use this client extension to override the payables invoice attributes.

See Also

Defining Provider and Receiver Controls, *Oracle Projects Implementation Guide*

Description

The extension is identified by the following items:

Item	Name
Body template	PACCINPB.pls
Specification template	PACCINPS.pls
Package	PA_CC_AP_INV_CLIENT_EXTN
Procedure	override_exp_type_exp_org

Table 9 – 64 Internal Payables Invoice Attributes Override Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Prerequisites

Complete the following actions before you use this extension:

- Complete all the implementation steps for cross charge and intercompany billing. See Oracle Projects Implementation Checklist, *Oracle Projects Implementation Guide*.

- Run the cost distribution process for the new transactions or use the Expenditure Items window to perform cross charge adjustments on existing transactions. Both processes identify cross charge transactions.
- Run the PRC: Generate Intercompany Invoices process to create receivables invoices for transactions that require intercompany processing.
- Run the PRC: Interface Intercompany Invoices to Receivables process to interface the intercompany invoices to Oracle Receivables.
- Run the PRC: Tieback Invoices from Receivables process to tie back the receivables invoices and create the internal payables invoices.

Parameters

The extension uses the following parameters:

Parameter	Usage	Type	Description
P_internal_billing_type	IN	VARCHAR2	Internal billing type– determines if the internal payables invoice is created for intercompany billing invoice or inter–project billing invoice. Valid values are: PA_IC_INVOICES (intercompany billing invoice) or PA_IP_INVOICES (inter–project billing invoice).
P_project_id	IN	NUMBER	Identifier of the provider project of an Inter–project billing invoice, or the intercompany billing project for an intercompany billing invoice.
P_receiver_project_id	IN	NUMBER	Identifier of the receiver project.

Table 9 – 65 (Page 1 of 4) Internal Payables Invoice Attribute Override Extension Parameters

Parameter	Usage	Type	Description
P_receiver_task_id	IN	NUMBER	Identifier of the receiver task for an inter–project billing invoice, or the task defined as the intercompany non–recoverable tax receiving task for an intercompany billing invoice.
P_draft_invoice_number	IN	VARCHAR	Number of the corresponding intercompany invoice for which the internal Payables invoice is created
P_draft_invoice_line_num	IN	NUMBER	Line number of the corresponding intercompany invoice line for which the internal payables invoice line is created
P_invoice_date	IN	DATE	Internal Payables invoice date
P_ra_invoice_number	IN	VARCHAR	Number of the internal Payables invoice
P_provider_org_id	IN	NUMBER	Identifier of the provider operating unit
P_receiver_org_id	IN	NUMBER	Identifier of the receiver operating unit
P_cc_ar_invoice_id	IN	NUMBER	Identifier of the corresponding invoice created in Receivables for the inter–company invoice
P_cc_ar_invoice_line_num	IN	NUMBER	Line number of the corresponding invoice line created in Receivables for the intercompany invoice

Table 9 – 65 (Page 2 of 4) Internal Payables Invoice Attribute Override Extension Parameters

Parameter	Usage	Type	Description
P_project_customer_id	IN	NUMBER	Identifier of the provider project customer for an inter–project billing invoice, or the intercompany billing project customer for an intercompany billing invoice
P_vendor_id	IN	NUMBER	Identifier of the supplier of the provider operating unit, based on the user setup for the provider operating unit in the Receiver Controls tab of the Provider / Receiver Controls window
P_vendor_site_id	IN	NUMBER	Identifier of the supplier site of the provider operating unit, based on the user setup for the provider operating unit in the Receiver Controls tab of the Provider / Receiver Controls window
P_expenditure_type	IN	VARCHAR2	Expenditure type for internal invoice distribution lines (Receiver Controls tab of the Provider / Receiver Controls window)
P_expenditure_organization_id	IN	NUMBER	Expenditure organization for internal invoice distribution lines (Receiver Controls tab of the Provider / Receiver Controls window)
X_expenditure_type	OUT	VARCHAR2	Expenditure type for the internal invoice distribution lines (determined by the client extension)

Table 9 – 65 (Page 3 of 4) Internal Payables Invoice Attribute Override Extension Parameters

Parameter	Usage	Type	Description
X_expenditure_organization_id	OUT	NUMBER	Expenditure organization for the internal invoice distribution lines (determined by the client extension)
X_status	OUT	NUMBER	Status indicating whether an error occurred. Valid values are: =0 Success <0 Oracle Error >0 Application Error
X_error_stage	OUT	VARCHAR2	Error handling stage
X_error_code	OUT	NUMBER	Error handling code

Table 9 – 65 (Page 4 of 4) Internal Payables Invoice Attribute Override Extension Parameters

Validation

The system performs the following validations:

- The value of x_expenditure_type must be a valid expenditure type for the expenditure type class of the supplier invoice.
- The value of x_expenditure_organization_id must be a valid expenditure organization for the receiver operating unit.

CHAPTER

10

Oracle Project Billing Client Extensions

This chapter describes the client extensions in the Oracle Project Billing application.

Funding Revaluation Factor Extension

Use the Funding Revaluation Factor Client Extension to apply a funding revaluation factor to the funding backlog amount. This extension may also be used to implement escalation indices defined for a contract. The factor can increase or decrease the funding backlog amount subject to revaluation and is applied to the funding backlog amount in the funding currency.

The client extension is called for each project or task by agreement.

Writing the Funding Revaluation Extension

The extension is identified by the following items:

Item	Name
Body template	PAXBFRCB.pls
Specification template	PAXBFRCS.pls
Package	Pa_Client_Extn_Funding_Reval
Procedure	Funding_Revaluation_factor

Table 10 – 1

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

The following table lists the parameters for the funding revaluation extension procedure:

Parameter	Usage	Type	Description
P_Project_ID	IN	NUMBER	Identifier of the project for which the revaluation process will be run. Value must be supplied.
P_Top_Task_ID	IN	NUMBER	Top task ID

Table 10 – 2 (Page 1 of 3)

Parameter	Usage	Type	Description
P_Agreement_ID	IN	NUMBER	Identifier of the agreement for which the revaluation process will be run
P_Funding_Currency	IN	VARCHAR2	The funding currency code
P_Projectfunc_Currency	IN	VARCHAR2	The project functional currency code
P_Invoiceproc_Currency	IN	VARCHAR2	The invoice processing currency code
P_Reval_Through_Date	IN	DATE	The revaluation through date
P_Reval_Rate_Date	IN	DATE	The rate date for the revaluation
P_Projfunc_Rate_Type	IN	VARCHAR2	The rate type of the project functional currency
P_Reval_Projfunc_Rate	IN	NUMBER	The project functional rate for the revaluation
P_Invproc_Rate_Type	IN	VARCHAR2	The rate type of the invoice processing currency
P_Reval_Invproc_Rate	IN	NUMBER	The revaluation rate for the invoice processing currency
P_Funding_Backlog_Amount	IN	NUMBER	The funding backlog amount
P_Funding_Paid_Amount	IN	NUMBER	The funding paid amount
P_Funding_Unpaid_Amount	IN	NUMBER	The unpaid funding amount
P_Projfunc_Backlog_Amount	IN	NUMBER	The backlog amount in project functional currency
P_Projfunc_Paid_Amount	IN	NUMBER	The paid amount in project functional currency
P_Projfunc_Unpaid_Amount	IN	NUMBER	The unpaid amount in project functional currency
P_Invproc_Backlog_Amount	IN	NUMBER	The backlog amount in invoice processing currency

Table 10 – 2 (Page 2 of 3)

Parameter	Usage	Type	Description
X_Funding_Reval_Factor	OUT	NUMBER	The funding revaluation factor. This factor is applied to the backlog funding amount.
X_Status	OUT	NUMBER	Displays the status of the process. Indicates whether the process had errors or not.

Table 10 – 2 (Page 3 of 3)

Billing Cycle Extension

You can use a billing cycle client extension to derive the next billing date for a project. To use a client extension, you must write the logic in a PL/SQL procedure and then store the procedure in the database.

To use the billing cycle extension for any project, you must set the project's Billing Cycle Type to *User–Defined.*

> **Note:** If a billing cycle extension used in the Invoice Generation Process returns a NULL value for the next billing date, the project will not be picked up for Invoice Processing.

Writing the Billing Cycle Extension

The extension is identified by the following items:

Item	Name
Body template	PAXIBCXB.pls
Specification template	PAXIBCXS.pls
Package	pa_client_extn_bill_cycle
Procedure	get_next_billing_date

Table 10 – 3

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Warning: Do not use the PL/SQL commands Commit and Rollback in your billing extension code. For the *get_next_billing_date* function, define the pragma RESTRICT_REFERENCES as WNDS, WNPS. For more information, refer to the *PL/SQL User's Guide and Reference Manual.*

Package.Function

pa_client_extn_bill_cycle.get_next_billing_date

The following table lists the parameters that Oracle Projects provides for the billing cycle client extension. The function returns a value for the next billing date.

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	The identifier of the project.
X_project_start_date	IN	DATE	The start date of the project.
X_billing_cycle_id	IN	NUMBER	The identifier of the billing cycle code.
X_bill_thru_date	IN	DATE	The bill–through date entered for the process.
X_last_bill_thru_date	IN	DATE	The last bill–through date of the project.

Table 10 – 4 (Page 1 of 1) Billing Cycle Extension Parameters

Billing Extensions

Billing extensions allow you to implement company–specific business rules to create automatic revenue and billing events. The Billing Extensions window requires you to specify either an amount or percentage when you assign the extension to a project type, project, or task.

These fields can be used as parameters in the billing extensions. The values for the parameters are available in the view PA_BILLING_EXTN_PARAMS_V. This view contains a single row that has all the conversion attributes used in the billing extension procedures.

With billing extensions, you can automatically calculate summary revenue and invoice amounts during revenue and invoice generation based on unique billing methods. These billing amounts are accounted for using events. Some examples of billing extensions you can implement are:

- Fee
- Surcharge
- Retention

This essay describes the implementation steps of billing extensions, as well as the processing of billing extensions and automatic events within Oracle Projects.

We also provide you with detailed information about designing and writing billing extensions, including information about public procedures and views you can use in your billing extensions to derive additional information. Finally, we provide you with information to help you test and debug billing extensions.

Warning: The public procedures and views in the Oracle Projects billing extensions are intended for use **only** in billing extensions for the Generate Draft Revenue / Generate Draft Invoice process. These public procedures and views will not work standalone or in any other client extensions.

Warning: Do not use the PL / SQL commands Commit and Rollback in your billing extension code.

Overview of Billing Extensions

To use the billing extension functionality, you must implement billing extensions and assign them to projects. Oracle Projects processes active billing extensions and accounts for the calculated revenue and invoice amounts.

Implementation

To implement your company–specific billing methods, you first design and write rules to calculate billing amounts using PL/SQL procedures. You then enter the billing extension definition in Oracle Projects to specify additional information (such as the procedure name to call) that is used by the revenue and invoice programs to process the extension.

Assignments

You define billing extensions in the Billing Assignments window and specify whether an amount or percentage is required for the extension when assigning the extension to a project type or task. Along with the amount and percentage, you can specify the currency and conversion attributes.

The values entered in the Billing Assignments window can be used in your billing extension by accessing the view pa_billing_extn_params_v. This view, which has a single row with all the conversion attributes, can be used to create multi–currency events with this extension. If you have custom code in your billing extension and want to use the parameters, you must update the extemsion.

.

Budget Type

You can specify which budget type to use as input to calculations that use budgeted amounts. If no value is given for budget type, the billing extension uses the Approved Cost Budget and/or Approved Revenue Budget. See: Retrieving Budget Amounts: page 10 – 33.

Processing

When you run the revenue or invoice processes, Oracle Projects looks for active billing assignments. When an assignment is found, the processes read the billing extension definition and call the appropriate procedure. If there are multiple active assignments for a project or

task, Oracle Projects calls the extension in ascending order based on the processing order specified in the billing extension definition.

Oracle Projects executes top task level assignments once for each top task. Billing extensions assigned to the project and the project type are executed once for each project, except in the case of task level funding. If a project uses task level funding, Oracle Projects executes billing extensions assigned to the project and the project type, once for each authorized top task on the project.

Automatic Events

Your billing extension calculates revenue and invoice amounts and creates one or more *Automatic* events to account for the revenue and invoice amounts. Oracle Projects processes these events as it does other manually entered events. You can store audit amounts for these events in the audit columns of the Events table.

Automatic events are events having an event type classification of *Automatic*. With automatic events, you can increase or decrease revenue and invoice amounts. You can also independently specify revenue and invoice amounts for the events. If an event has both a nonzero revenue amount and a nonzero invoice amount, you must use the same sign for both amounts. Some examples of revenue and invoice amounts for these events are:

- Revenue = $100, Invoice = $0
- Revenue = $100, Invoice = $200
- Revenue = –$100, Invoice = –$100
- Revenue = $0, Invoice = –$100

The billing extension uses the public procedure pa_billing_pub.insert_event to create automatically created events. This is shown in the following table.

Item	Name
Body template	PAXITMPB.pls
Specification template	PAXITMPS.pls

Table 10 – 5

Item	Name
Package	pa_billing_pub
Procedure	insert_events

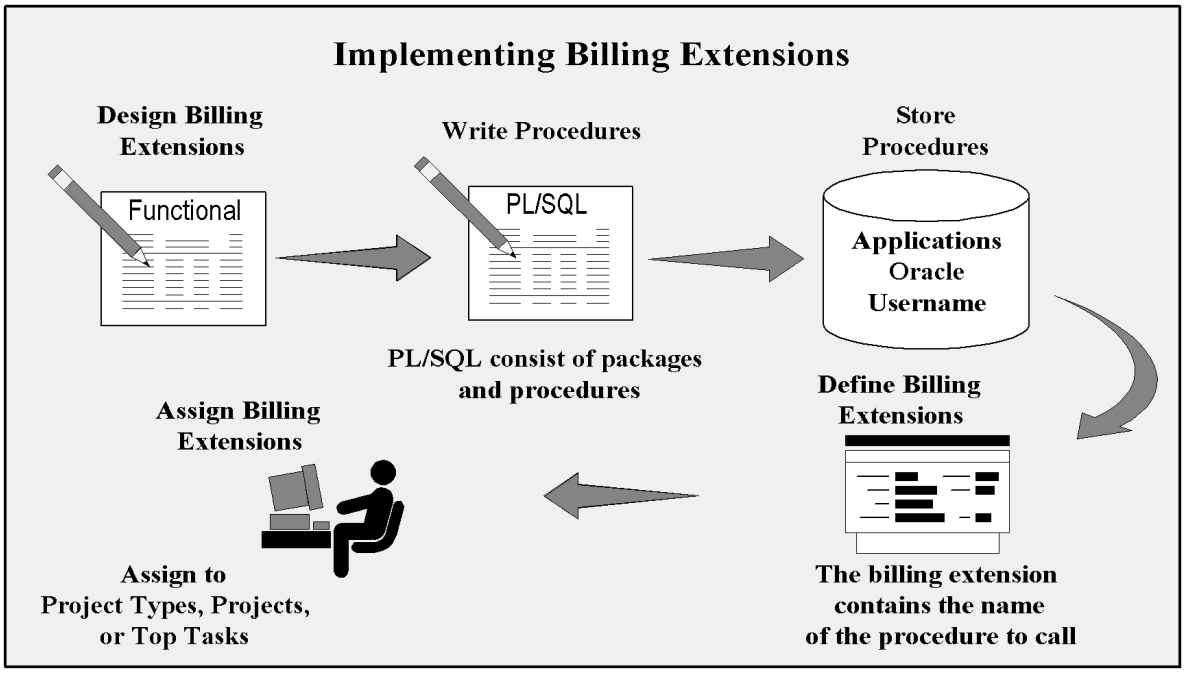
Table 10 – 5

See Also

Event Types, *Oracle Projects Implementation Guide*

Overview of Implementation Steps

Figure 10 – 1



To implement billing extensions in Oracle Projects according to your company's method of doing business, perform the following steps.

Step 1 Design billing extensions

Carefully plan the definition of billing extensions before you begin writing them. Typically, the logic of your billing extensions are dependent on your company's implementation of Oracle Projects. Consider the following issues when designing your billing extensions:

- Logic of billing extensions
- Additional implementation data required

Step 2 Write and store PL/SQL procedures

After you design your billing extensions, write the PL/SQL procedures that define the logic of the billing extensions.

After you write your procedures, store them in the database and test them to ensure that your billing extension logic works as expected.

Step 3 Define billing extensions

Define your billing extensions, which specify the PL/SQL procedure name and additional information for Oracle Projects to use when processing billing extensions.

You use the Billing Extensions window to define billing extensions.

This step assumes that an event type has already been defined for the default event type. For a discussion of automatic events created by billing extensions, see: Automatic Events: page 10 – 9.

Step 4 Assign billing extensions to project types

Assign billing extensions to the appropriate project types if you have defined non–project–specific billing extensions. Your project users will assign the project–specific billing extensions to projects and top tasks as they define projects.

You use the Project Types form to assign billing extensions to project types.

See Also

Designing Billing Extensions: page 10 – 12

Writing Billing Extension Procedures: page 10 – 26

Defining Billing Extensions: page 10 – 40

Defining Project Types, *Oracle Projects Implementation Guide*

Designing Billing Extensions

Before you begin designing billing extensions, you should familiarize yourself with the three classes of billing extensions to understand the complexity of the business problem you are trying to solve.

There are also specific questions of client extension design that are unique to determining the requirements and logic of your billing

extensions. We list these questions in the pages that follow, and then address some of these issues in further detail in the Concepts of Billing Extension Definitions section: page 10 – 15.

Understanding Billing Extensions Classes

There are three primary classes of billing extensions that you can write. The classes differ by how you calculate the revenue and invoice amounts:

Class 1: Revenue and Invoice Amounts

This class of billing extensions is based on a function of the revenue and invoice amounts included on draft revenue and invoices

An example is a Surcharge billing extension, which is typically a percentage of the invoice amount. This is the simplest class of billing extension to design and write.

Class 2: Independent Values

This class of billing exctensions is based on values independent of the amounts included on draft revenue and invoices.

An example is the percent complete revenue accrual method, which is based on the physical percent complete entered for the project multiplied by the budget revenue amount. The calculated amount is independent of other amounts included on the revenue and invoice. In many cases, this class of billing extensions may be the only method used to calculate revenue and invoice amounts for the project, particularly if you are using Event based revenue accrual and invoicing.

Class 3: Transaction Attributes

This class of billing extensions is based on the attributes of a group of transactions included on draft invoices, for which the billing extension calculates the amount to bill for these transactions.

For example, you may wish to calculate the revenue and invoice amounts based on number of days worked, rather than the actual hours worked which are recorded on the timecard. Another example is volume discounts on an invoice, in which you provide discounts based on the volume of transactions billed. You calculate the amount to bill for the group of transactions without specifying a bill amount for each transaction.

To properly track which individual transactions are billed using an automatic event, you must set up your projects to include these transactions on an invoice, but without an invoice amount. These transactions must have a nonzero revenue amount and a invoice amount of zero. Oracle Projects includes these transactions on the invoice on a net zero adjustment line which you cannot review in the forms, but that you can read from the database in your billing extension. You can set up a project to process transactions in this way by using different revenue and invoice burden schedules; the revenue schedule determines the appropriate revenue amounts and the invoice schedule calculates a invoice amount equal to zero.

Oracle Projects links the detail transactions to the invoice on a net zero adjustment invoice line, and you hold and account for the summary bill amount for these transactions using an automatic event included on the invoice. You can then write custom reports to list the detail transactions that backup the summary event amount.

You can only implement this class of billing extensions for invoicing amounts. You cannot use this class for revenue amounts calculated during revenue generation.

Planning Your Billing Extensions

You should carefully design billing extensions before implementing them in Oracle Projects. Careful planning of your billing extension helps to ensure that you are calculating and accounting for revenue and invoice amounts according to your company–specific rules. See: Designing Client Extensions: page 7 – 6.

You should consider the additional design issues for billing extensions:

- Are you calculating a revenue amount, an invoice amount, or both? Are the amounts generated during revenue accrual, invoice generation, or both?
- How are the amounts calculated? What are the inputs to the calculation?
- How are the inputs derived?
- How are the amounts processed: (1) for reporting purposes (2) for accounting purposes, (3) for invoicing?
- How are the attributes of the automatic event set: event type, event organization, event description, completion date?
- Under what conditions is this calculation used? What types of projects? What types of billing terms?

- How is the billing extension processed for adjustments? Adjustments are defined as revenue credits or invoice credit memos, based on other transactions.
- Can this billing extension be called with other billing extensions on the same project / task? If so, what is the dependency and order of your billing extensions?
- What is the exception handling if some input values cannot be found?
- How is the logic affected if the inputs change over time?
- Is there a limit on the amount calculated? If so, what is the logic?
- Are there implications of the level at which the project is funded – either the project level or the top task level? If so, what are they?

Once you answer these questions, you should have the appropriate information to define a billing extension in Oracle Projects and to document the functional specifications for your technical resource to use in writing the PL / SQL procedure.

Concepts in Billing Extension Definitions

When you enter billing extension definitions, you specify parameters that specify how your billing extension is processed in Oracle Projects. This section explains some of these parameters.

Calling Process

You specify if the billing extension is called by the revenue generation program, the invoice generation program, or both programs.

When you call billing extensions during revenue generation, you can create events with a revenue amount, or with a revenue amount and a bill amount, as long as the revenue amount is nonzero.

When you call billing extensions during invoice generation, you can create events with a bill amount, or with a revenue amount and a bill amount, as long as the bill amount is nonzero.

The following table shows examples of events with various revenue and bill amounts that you can create in the Generate Draft Revenue calling process.

Billing Extension	Event	Revenue Amount	Bill Amount	Comments
1	1	100	100	Bill amount is not processed until revenue for the event is distributed
2	2	100	–	Revenue event only

Table 10 – 6 Calling Process: Example Events (Page 1 of 1)

The following table shows examples of events that you can create in the Generate Draft Invoice calling process.

Billing Extension	Event	Revenue Amount	Bill Amount	Comments
3	1	100	100	Revenue amount is not processed until the invoice on which the event is billed is released
4	2	–	100	Invoice event only

Table 10 – 7 Calling Process: Example Events (Page 1 of 1)

If you create an event with both revenue and bill amounts, the revenue amount and the bill amount do not have to be the same. You can create positive or negative event amounts with billing extensions.

You can create a billing extension that is called by both revenue generation and invoice generation. You would do this if your billing calculation is similar for both the revenue and bill amounts, with the exception that the event revenue amount is based on the accrued revenue, and the event bill amount is based on the amount invoiced. You can write your procedure to have the same logic for the calculation but to use the appropriate input of either accrued revenue or amount invoiced into your calculation. With this approach of writing one procedure and one billing extension, you can avoid duplication of your logic. In addition, your project users only need to assign one billing extension to their projects, instead of two billing extensions – one for revenue accrual and one for invoicing.

Calling Place: Revenue Generation Program

The revenue generation program calls client extensions during the following three processing steps:

- Revenue Deletion Processing
- Revenue Adjustment Processing
- Revenue Regular Processing

Revenue Deletion Processing

During revenue deletion, calls are made to the following billing extensions, in the order shown below:

- PRE Billing Extension
- DEL Billing Extension

Standard revenue processing is then performed, followed by the following billing extension call:

- POST Billing Extension

Revenue Adjustment Processing

During revenue adjustment, calls are made to the following billing extensions, in the order shown below:

- PRE Billing Extension

Standard adjustment revenue processing is then performed, followed by the following billing extension calls:

- ADJ Billing Extension
- POST Billing Extension

Regular Revenue Processing

During normal revenue processing, calls are made to the following billing extensions, in the order shown below:

- PRE Billing Extension

Regular revenue processing is then performed, followed by the following billing extension call:

- REG Billing Extension

Automatic revenue event processing is then performed, followed by the following billing extension call:

- POST–REG Billing Extension

Automatic revenue event processing is performed again, followed by the following billing extension call:

- POST Billing Extension

Calling Place: Invoice Generation Program

The invoice generation program calls client extensions during the following three processing steps:

- Invoice Deletion Processing (when using the delete & regenerate option only)
- Invoice Cancellation Processing (when using the cancel option only)
- Invoice Write–Off Processing (when using the write–off option only)
- Invoice / Credit Memo (Regular) Processing

Invoice Deletion Processing

During invoice deletion, when the delete and regenerate option is used, calls are made to the following billing extensions, in the order shown:

- Call PRE Billing Extension
- Call DEL Billing Extension

Standard delete invoice processing is then performed, followed by the following billing extension call:

- Call POST Billing Extension

Invoice Cancellation Processing

During invoice cancellation, when the cancel option is used, calls are made to the following billing extensions, in the order shown below:

- PRE Billing Extension

Standard delete invoice processing is then performed, followed by the following billing extension calls:

- CANCEL Billing Extension
- POST Billing Extension
- Approval / Release Billing Extension

Invoice/Credit Memo (Regular) Processing

During invoice and credit memo (regular) processing, calls are made to the following billing extensions, in the order shown below:

- PRE Billing Extension

Standard credit memo processing is then performed, followed by the following billing extension calls:

- ADJ Billing Extension
- POST Billing Extension
- PRE Billing Extension

Regular invoice processing is then performed, followed by the following billing extension call:

- REG Billing Extension

Automatic invoice event processing is then performed, followed by the following billing extension calls:

- POST–REG Billing Extension
- Approval / Release Billing Extension
- POST Billing Extension
- Validate Approval / Release Processing

Standard write–off invoice processing is then performed.

Extension Call Types

There are several predefined places within the revenue generation and invoice generation programs where your billing extension can be called when processing a project:

- Pre–Processing
- Delete Processing
- Cancel Invoice Processing
- Write–Off Invoice Processing
- Adjustment Processing
- Regular Processing
- Post–Regular Processing
- Post–Processing

Pre–Processing	*Pre–processing* billing extensions are called before any revenue accrual or invoice calculations for a project. The Generate Draft Revenue and Generate Draft Invoices processes do not allow you to create automatic events in this calling place. An example of a preprocessing billing extension is to place all unbilled, unpaid supplier invoice items on hold, so that they are not billed; and to release the billing hold on any unbilled, paid supplier invoice transactions that are on hold. You can then bill the paid supplier invoice items during standard invoice processing.
Delete Processing	*Delete processing* billing extensions are called after revenue is billed and before any revenue accrual or invoice calculations for a project; this is only applicable to invoicing billing extensions The Generate Draft Invoices process does not allow you to create automatic events in this calling place.
Cancel Invoice Processing	*Cancel invoice processing* billing extensions are called after the invoice cancellation for a project. This is only applicable to invoice billing extensions. The Generate Draft Invoices process does not allow you to create automatic events in this calling place.
Write–Off Invoice Processing	*Write–off invoice processing* billing extensions are called after the invoice write–off processing for a project. This is only applicable to invoice billing extensions. The Generate Draft Invoices process does not allow you to create automatic events in this calling place.
Adjustment	*Adjustment* processing creates crediting revenue and invoices that credit existing revenue or invoices. Oracle Projects creates crediting revenues and invoices due to changes in revenue or invoice amounts or in the revenue general ledger account. These credits are created for one or more individual transactions which have previously been processed and included on a draft revenue or invoice; these changes in amounts or accounts result from adjustment actions on the individual transactions. You can create automatic events in this step. If you transfer these events to Oracle Receivables for autoinvoicing, link the automatic event invoice lines to their corresponding events in the original invoice. See: Inserting Events: page 10 – 29. Oracle Projects calls a billing extension in this step after all of the crediting revenue and invoices are created.
Regular	*Regular* processing creates non–crediting revenue and invoices. Oracle Projects creates revenue and invoices based on individual transactions and events that have not previously been processed for revenue accrual and invoicing.

You can create automatic events in this step. Oracle Projects calls a billing extension in this step after all non–crediting revenues and invoices are created.

Post–Regular

Post–regular processing billing extensions create events based on all prior revenue generated in order to base the calculation on the total revenue accrued, including other automatic events. An example of a post–regular processing billing extension is cost accrual based on the revenue generated. See: Revenue–Based Cost Accrual, *Oracle Project Billing User Guide*.

Post–Processing

Post–processing billing extensions are called after all of the adjustment, regular, and post–regular processing is complete. The Generate Draft Revenue and Generate Draft Invoices processes do not allow you to create automatic events in this calling place. All of the revenue and invoice processing is complete before this step is executed. An example of a post–processing billing extension is to notify a project manager when an invoice greater than $25,000 is created.

The following table shows an example of the different automatic events created by using different calling places for a billing extension based on a percentage of the amount invoiced.

Period	Invoice Number	Invoice Credited	Invoice Amount	Automatic Event Amount (Regular and Adjustment)	Automatic Event Amount (Regular Only)
1	1		1000	100	100
2	2	1	–500	–50	
	3		1500	150	100
Summary:			2000	200	200

Table 10 – 8 Calling Place: Example Events (Page 1 of 1)

The billing extension called only during regular processing accounted for the total amount invoiced, including the credited amount during regular processing as illustrated by the event created for invoice number three.

Transaction Independent

Once you determine the inputs to your calculations, you can determine if your billing extension calculation is solely dependent on other transactions being processed, or if your calculation can be executed without any other transactions being processed. Transactions refer to expenditure items and events.

Transaction independent billing extensions are executed for each project with an active billing assignment, even if there are no transactions to process. This type of billing extension relies on an input other than billable transactions on a project. If this input changes, the calculated billing amount changes, which you want to record. For example, the cost–to–cost revenue accrual method, which relies on the budgeted cost and revenue amounts. If the budgeted cost or budgeted revenue changes, the revenue amount changes. You want to record this revenue amount change even if no other transactions are processed in revenue generation. This category includes the class of billing extensions that calculate revenue and invoice amounts based on values independent of the amounts included on draft revenue and invoices.

> **Note:** If you design a billing extension to be transaction independent, it will be executed in every run of the revenue or invoice processes.

Transaction dependent billing extensions are executed only if there are other transactions processed. An example of this type of billing extension is surcharge in which you calculate a percentage of the amount billed. You do not want to bill surcharge if no other transactions are billed.

Transaction dependent billing extensions are called only if billable expenditure items and events exist that need to be processed. For example, there may be new transactions that are set to Non–Billable, which are not going to generate any revenue or bill amount and will not cause the billing extension to be called. This category includes billing extensions that calculate revenue and invoice amounts based on (i) a function of the revenue and invoice amounts included on draft revenue and invoices, or (ii) the attributes of a group of transactions included on draft invoices.

The following table shows an example of transaction dependent and transaction independent billing extensions. Billing extension 1, which is transaction dependent, calculates 10% of the invoice amount. Billing extension 2, which is transaction independent, bills $100 per period regardless of amount invoiced in that period.

Period	Invoice Number	Invoice Credited	Invoice Amount	Automatic Event Amount (Transaction Dependent)	Automatic Event Amount (Transaction Independent)
1	1		1000	100	100
2	2	1	–500	–50	
	3		1500	150	100
3	4		–	–	100
Summary:			2000	200	300

Table 10 – 9 Transaction Independent Example: Example Events (Page 1 of 1)

Relationship between Calling Place and Transaction Independent

The parameters for calling place and transaction independent are related.

You should call any transaction dependent billing extension in both regular and adjustment processing. This will ensure that all adjustments, including those that do not result in a new non–crediting amount, are properly accounted for. For example, you may have a non–billable adjustment which reverses amounts, but does not process any new non–crediting amounts.

You only need to call your transaction independent billing extension once during processing for a project, which can be done during regular processing. You typically do not call transaction independent billing extensions during adjustment processing.

The table below summarizes how you should set up the calling place and transaction independent parameters in your billing extension definition, based on the type of billing extension calculation.

Billing Extension Calculation	Regular	Adjustment	Transaction Independent
Based solely on transactions	Yes	Yes	No
Based on inputs other than transactions	Yes	No	Yes

Table 10 – 10 Calling Place and Transaction Independent Parameters (Page 1 of 1)

There are exceptions to the general rule shown in the above table. You may define a billing extension as transaction dependent, but to be called only during regular processing. For example, you want to charge interest on outstanding invoices, but only want to include the interest on an invoice that has other transactions included on it. The interest calculation itself is a transaction independent calculation, but you define it as transaction dependent so that it is calculated only when other transactions are processed for an invoice. You do not want to create invoices with only an interest amount.

Project–Specific

You need to determine if your billing extension implements a company policy across projects or if it is applicable only to specific projects for which it is negotiated.

Project–specific billing extensions are those methods which are applicable only to specific projects for which they are negotiated. Project users assign these billing extensions to projects and top tasks; you cannot assign these billing extensions to project types.

Non–project–specific billing extensions are those methods which implement company policy across projects. You assign these billing extensions to project types; the billing extension applies to all projects of that project type. Project users cannot assign these billing extensions to projects.

Suggestion: You can include conditional logic in your procedure to allow exceptions to project type rules.

Event Attributes

When designing billing extensions, you can specify the attributes of automatic events that are created by billing extensions. You can use the following default values or override the defaults for any of these attributes.

Event Attribute	Comments
Event Type	Defaults to event type on billing extension; event type must have an event type classification of A*utomatic*.
Event Description	Defaults to event description on billing extension.
Event Organization	Defaults to managing organization of project or task to which the event is assigned.
Completion Date	Accrue through date for events created during revenue generation, bill through date for events created during invoice generation.
Revenue Amount	For billing extensions called in revenue generation, must specify revenue amount. For billing extensions called in invoice generation, can optionally specify revenue amount; revenue amount is not processed until invoice on which the event is billed is released.
Bill Amount	For billing extensions called in invoice generation, must specify bill amount. For billing extensions called in revenue generation, can optionally specify bill amount; bill amount is not processed until revenue for the event is accrued.
Descriptive Flexfield Segments	Can pass any value as long as the value is valid with the descriptive flexfields you have defined for events.
Audit Columns in Events	For values used in billing extension calculations. NOTE: not displayed to the user, but available in the table. See: Insert events: page 10 – 29.

Table 10 – 11 (Page 1 of 1) Attributes of Automatic Events

Budget Attributes

When designing billing extensions, you can specify the attributes of budgets that are used by billing extensions. You can use the following default values or override the defaults for any of these attributes.

Budget Attribute	Comments
Cost Budget Type Code	Defaults to Approved Cost Budget.
Revenue Budget Type Code	Defaults to Approved Revenue Budget.

Table 10 – 12 (Page 1 of 1) Attributes of Budgets

Writing Billing Extension Procedures

Oracle Projects revenue and invoice generation programs call your billing extension procedures which define the logic to calculate and create automatic events according to your rules.

Your procedure can call other procedures or views. You can use predefined procedures and views, or you can write your own procedures. We discuss these predefined procedures and views in more detail in the pages that follow.

Procedure Template

The extension is identified by the following items:

Item	Name
Body template	PAXITMPB.pls
Specification template	PAXITMPS.pls

Table 10 – 13

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

The following table lists the parameters that Oracle Projects provides for the billing extension procedure.

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	Identifier of the project of the billing assignment.
X_top_task_id *	IN	NUMBER	Identifier of the top task of the billing assign. This parameter has a value in the following cases: 1) Billing extension assigned to top task 2) Project for which a billing extension is applicable is funded at the top task level. The billing extension is executed once for each authorized top task belonging to the project.
X_calling_process	IN	VARCHAR2	Specifies whether the revenue or invoice program is calling the billing extension. The possible values of this parameter are *Revenue* or *Invoice*.
X_calling_place	IN	VARCHAR2	Specifies where the billing extension is called in the revenue or invoice program. Possible values are PRE, POST, REG, or ADJ.
X_amount	IN	NUMBER	The amount entered on the billing assignment.
X_percentage	IN	NUMBER	The percentage entered on the billing assignment.
X_rev_or_bill_date	IN	DATE	Specifies the accrue through date if called by revenue generation, or the bill through date if called by invoice generation.
X_bill_extn_ assignment_id	IN	NUMBER	ID of the billing assignment being processed. Use this to select information (such as descriptive flexfield values) from the billing assignment.
X_bill_extension_id	IN	NUMBER	ID of the billing extension being processed. Use this to select information (such as descriptive flexfield values) from the billing extension definition.
X_request_id	IN	NUMBER	Request ID of the current run.

Table 10 – 14 (Page 1 of 1) Billing Extension Parameters

Note: * You cannot create project level events for projects using task level funding. You must write your billing extensions so that they work if they are called with or without this parameter.

Views and Procedures You Can Use

Oracle Projects provides public, predefined procedures and views that you can use within your billing extension procedures for the Generate Draft Revenue and Generate Draft Invoice processes to derive amounts and create events. These procedures are created in a package named pa_billing_pub.

> **Note:** You cannot use the public billing extension procedures or views by themselves or from any other client extension.

In the pages that follow, we provide you with a description of each procedure, information about the parameters available for the procedure, and any additional information you need to use the procedure in your billing extension. Use these procedures and views to:

- Calculate amounts: page 10 – 28
- Identify transactions processed in the current run: page 10 – 28
- Insert events: page 10 – 29
- Retrieve budget amounts: page 10 – 33
- Handle error conditions: page 10 – 35

Calculating Amounts

Oracle Projects provides two views that you can use to identify detail expenditure items included on draft revenue and draft invoices processed in a given run. Use these views in your calculations for transaction dependent billing extensions. The views display the detail transactions processed for the context in which a billing extension is called, which consists of a project, a top task (if task level assignment), a calling place, and a request ID.

- PA_BILLING_REV_TRANSACTIONS_V (use this in procedures that are called during revenue generation)
- PA_BILLING_INV_TRANSACTIONS_V (use this in procedures that are called during invoice generation)

Identifying Process Run Information

Oracle Projects provides four views that you can use to identify the detail revenue and invoice transactions processed in the current run.

- PA_BILLING_REV_DELETION_V displays the draft revenues that will be deleted in the current draft revenue generation run.

Use this view in the billing extension called during the deletion processing of revenue generation.

- PA_BILLING_REV_INV_DELETION_V displays the draft invoices that will be deleted in the current draft revenue generation run. Use this view in the billing extension called during the deletion processing of revenue generation.
- PA_BILLING_INV_DELETION_V displays the draft invoices that will be deleted in the current draft invoice generation run. Use this view in the billing extension called during the deletion processing of invoice generation.
- PA_BILLING_INV_PROCESSED_V displays the invoices that were processed in the current run.

Inserting Events

Use the insert_events procedure to create automatic events in the events table. You must use this procedure when creating events using billing extensions, as it contains validation that ensures the data integrity of the events that you create.

If this procedure encounters an error, it displays an error message in the log file of the process that called the procedure and does not create an event.

Package.Procedure

pa_billing_pub.insert_event

Listed below are the parameters available for the insert_event procedure.

Parameter	Usage	Type	Description
X_rev_amt	IN	REAL	Revenue amount of event. If the billing extension is called by revenue generation only, you must set the revenue amount to a nonzero number. If the billing extension is called by invoice generation, then if the bill amount is positive, the revenue amount must also be positive
X_bill_amt	IN	REAL	Bill amount of event. If the billing extension is called by revenue generation only, then if the revenue amount is positive, the bill amount must also be positive. .If the billing extension is called by invoice generation, you must set the bill amount to a nonzero number.
X_project_id	IN	NUMBER	ID of the project to which the event is assigned.
X_event_type	IN	VARCHAR2	Event type of event. If a default is not specified, you must provide a value. Event type must have an event type classification of AUTOMATIC.
X_top_task_id	IN	NUMBER	ID of the top task to which the event is assigned.
X_organization_id	IN	NUMBER	ID of event organization.
X_completion_date	IN	DATE	Completion date of event.
X_event_description	IN	VARCHAR2	Description of event. If you do not specify a default event description, you must provide a value here.
X_event_num_reversed	IN	NUMBER	Original automatic event number.
X_attribute_category	IN	VARCHAR2	Descriptive flexfield context.
X_attribute1–10	IN	VARCHAR2	Descriptive flexfield segments.
X_audit_amounts1–10	IN	NUMBER	Audit amounts for events.
X_audit_cost_budget_type_code	IN	VARCHAR2	Audit cost budget type code.
X_audit_rev_budget_type_code	IN	VARCHAR2	Audit revenue budget type code.
X_inventory_org_id	IN	NUMBER	Inventory organization ID

Table 10 – 15 Insert Event Parameters (Page 1 of 3)

Parameter	Usage	Type	Description
X_inventory_item_id	IN	NUMBER	Inventory item ID
X_quantity_billed	IN	NUMBER	Bill quantity
X_uom_code	IN	VARCHAR2	Unit of measure
X_unit_price	IN	NUMBER	Contract Price
X_reference1 through X_reference10	IN	VARCHAR2	Generic reference column
X_txn_currency_code	IN	VARCHAR2	Event transaction currency code
X_project_rate_type	IN	VARCHAR2	Exchange rate type used to convert from transaction currency to project currency
X_project_rate_date	IN	DATE	Exchange rate date used to convert from transaction currency to project currency
X_project_exchange_rate	IN	NUMBER	Exchange rate used to convert from transaction currency to project currency
X_project_func_rate_type	IN	VARCHAR2	Exchange rate type used to convert from transaction currency to project currency
X_project_func_rate_date	IN	DATE	Exchange rate date used to convert from transaction currency to project functional currency
X_project_func_exchange_rate	IN	NUMBER	Exchange rate used to convert from transaction currency to project functional currency
X_funding_rate_type	IN	VARCHAR2	Exchange rate type used to convert from transaction currency to funding currency
X_funding_rate_date	IN	DATE	Exchange rate date used to convert from transaction currency to funding currency
X_funding_exchange_rate	IN	NUMBER	Exchange rate used to convert from transaction currency to funding currency

Table 10 – 15 Insert Event Parameters (Page 2 of 3)

Parameter	Usage	Type	Description
X_error_message	OUT	VARCHAR2	Error message text.
X_status	OUT	NUMBER	Status indicating whether an error occurred. Valid values are: =0 Successful validation <0 Oracle error (message will be written into a log file) >0 Application error

Table 10 – 15 Insert Event Parameters (Page 3 of 3)

Business Rules:

- If the billing extension creates a new automatic event from a transaction adjustment, the billing extension looks for the original event number (X_event_num_reversed). If the billing extension finds no value, you will receive the error message "You must have specified original event number for ADJ automatic event."

  Note: Oracle Projects provides a view that you can use to identify to original automatic event information of the current project, top task, and the credited invoices of the current request:

  – PA_BILLING_ORIG_EVENTS_V

- Currency attribute rules:

  – The transaction currency code is passed only if the bill transaction revenue or bill amount parameter is populated.

  – If the transaction currency code, rate, and amounts are not passed to the procedure, the procedure uses the project functional currency code and amounts.

  – If the transaction currency is the same as the project functional currency, the procedure ignores the rate type, rate date, and rate.

  – If transaction currency is different from the project functional currency and currency attributes are not passed, the procedure will use project defaults.

Note: For a description of the currency conversion business rules, see: Setting Up Multi–Currency Billing, *Oracle Projects Implementation Guide*.

Predefined Billing Extensions

The billing transaction currency of automatic events that are created by the predefined Percent Complete or Cost Accrual billing extensions is project functional currency.

Retrieving Budget Amounts

Use the get_budget_amount procedure to retrieve baselined budgeted cost or revenue amounts for use in your calculations.

Package.Procedure

pa_billing_pub.get_budget_amount

Listed below are the parameters available for the get_budget_amount procedure. You must specify a value for the **X2_project_id** parameter for this procedure. You can optionally use the X2_task_id parameter to derive the budget amount for a task.

The parameters include input and output parameters for cost and revenue budget type codes.

Parameter	Usage	Type	Description
X2_project_id	IN	NUMBER	ID of project to retrieve baselined budget amounts.
X2_task_id	IN	NUMBER	ID of top task to retrieve current budget amounts; you must also specify the project ID when you use this parameter.
X2_revenue_amount	OUT	REAL	Baselined revenue budget amount for project or task.
X2_cost_amount	OUT	REAL	Baselined cost budget amount for project or task.

Table 10 – 16 (Page 1 of 2) Get Budget Amount Parameters

Parameter	Usage	Type	Description
p_cost_budget_ type_code	IN	VARCHAR2	Cost budget type code to be used for calculating cost budget amount. If this value is not specified, the cost budget type in the billing extension setup table is used. If no value is entered in the billing extension setup table, the Approved Cost Budget is used.
p_rev_budget_ type_code>	IN	VARCHAR2	Revenue budget type code to be used for calculating revenue budget amount. If this value is not specified, the revenue budget type in the billing extension setup table is used. If no value is entered in the billing extension setup table, the Approved Revenue Budget is used.
X_cost_budget_ type_code	OUT	VARCHAR2	Cost budget type code that was used for calculating the cost budget in this public API.
X_rev_budget_ type_code	OUT	VARCHAR2	Revenue budget type code that was used for calculating the revenue budget in this public API.
X_error_message	OUT	VARCHAR2	Error message text.
X_status	OUT	NUMBER	Status indicating whether an error occurred. Valid values are: =0 Successful validation <0 Oracle error (message will be written into a log file) >0 Application error

Table 10 – 16 (Page 2 of 2) Get Budget Amount Parameters

Error Handling

Use the insert_message procedure to create debugging and error messages in the PA_BILLING_MESSAGES table. When you encounter a problem with billing extensions, you can review these messages in the log file of the revenue and invoice processes that call the billing extension, or you can review the error message table.

Package.Procedure

pa_billing_pub.insert_message

Listed below are the parameters available for the insert_message procedure.

Parameter	Usage	Type	Description
X_inserting_ procedure_name	IN	VARCHAR2	Name of procedure that is inserting the message.
X_message	IN	VARCHAR2	The free text to display as the message in the log file.
X_attribute1–15	IN	VARCHAR2	Descriptive flexfield segments of billing message. These attributes appear in the billing messages table only.
X_error_message	OUT	VARCHAR2	Error message text.
X_status	OUT	NUMBER	Status indicating whether an error occurred. Valid values are: =0 Successful validation <0 Oracle error (message will be written into a log file) >0 Application error

Table 10 – 17 (Page 1 of 1) Insert Message Parameters

Additional Considerations for Writing Procedures

You should understand the following issues and determine how they affect your PL/SQL procedure.

Hard Limits and Automatic Events

Oracle Projects processes automatic events as it does manual events. When events are processed for a project that is at the hard limit, only those events that fully fit under the hard limit are processed. If the event amount does not fully fit under the hard limit, it is created but not processed on a draft revenue or invoice until there is enough funding available. Deleting the revenue does not delete the event; however, regenerating the revenue creates a new duplicate event. Once you raise the hard limit, Oracle Projects processes both events, which will lead to duplicate event amounts.

To avoid the creation of duplicate events, you can include logic in your billing extension to create an automatic event only if no unprocessed automatic events exist or if it will fit under the hard limit and be processed accordingly. Otherwise, the billing extension does not create the event, and you should delete the revenue without releasing it. If you do release the revenue, you need to calculate and insert the event manually.

In some transaction independent cases, you may wish to insert an amount that fits under the limit. In most transaction dependent cases, you should insert the entire amount, regardless of the limit to account for amounts based on processed transactions.

Suggestion: If you are creating positive and negative event amounts, create the negative amount first, so that it increases available funding.

Multiple Customers and Automatic Events

Oracle Projects processes automatic events as it does manual events. With multiple customer projects, events are split between the customers based on the customer billing percentage.

If you include hard limit logic in your procedure, you need to consider multiple customers and hard limit processing.

Creating Multiple Events in Same Calling Place in Same Run

It is possible for one or more billing extensions to create events in the same calling place in the same run. All billing extensions are executed in the calling place before any of the automatic events are included on the invoice or revenue. You need to consider the issues in the case in which one billing extension is dependent on the amount of other events processed in that calling place in the same run.

For example, assume you are processing a surcharge extension and a retention extension in the regular processing section of invoice generation. The surcharge is executed before the retention based on the processing order of the billing extension definition. The surcharge event is created but is not yet included on the invoice. The retention extension relies on the total invoice amount. To get the total invoice amount, the retention extension must account for the surcharge event which is not yet included on the invoice.

You must include logic in your billing extension to read any automatic event created for projects and tasks in the same run and calling place.

Tips on Writing and Debugging Procedures

You can make testing and debugging your billing extension procedure much easier by writing your procedure in a very methodical, structured approach as suggested below. Your functional and technical resources should work together to validate the billing extension.

Step 1 Create own billing extension to create event of a given amount

The first step is to create a very simple billing extension using the template files. You perform these steps to create an automatic event using a billing extension.

- Copy the template files to your own files
- Change the package and procedure names
- Add one call to the insert_event procedure to create an event of a given amount
- Store the procedure in the database
- Define a billing extension in Oracle Projects using this procedure
- Assign the billing extension to a *test* project
- Process the project through revenue and invoice generation; you should run the process that is appropriate for the billing extension
- Verify that an event is created for the given amount

Step 2 Test each SQL statement in SQL*Plus

After you verify that your billing extension works in an integrated flow, you can begin to build the logic of your billing extension. You first

write and test each SQL statement in SQL*Plus. You focus on each SQL statement independently until you have verified all of the SQL statements.

> **Note:** Be sure that the appropriate SQL statements handle both project level and top task level billing assignments.

If you are writing transaction dependent billing extensions, you should create the appropriate transactions on your test project and then process the transactions through revenue accrual or invoicing. Note the request ID of the process. All of the transactions are marked with this request ID, so you can use the request ID in testing your SQL statements in SQL*Plus. You can then use one of the following views to read the appropriate transactions processed by the request ID.

- PA_BILLING_REV_TRANSACTIONS_V
- PA_BILLING_INV_TRANSACTIONS_V

The views use PL/SQL functions, which are included in the view definition, to determine the appropriate project, task, calling place, and request ID variables for which the billing extension is being run. These variables are set by the revenue generation and invoice generation processes before the billing extension is executed. If you do not set these variables, then the view returns all records for that project and task in SQL*Plus. You can set these variables for your SQL*Plus session by running the papbglob.sql script. You can test your SQL statements using views with the variables that you want.

Step 3 Add SQL statements one at a time and test in an integrated flow

After you test and verify each SQL statement that you plan to use in your billing extension, you can add one SQL statement at a time to your billing extension definition. Each time after you add a new part of the logic to the billing extension, you should then test your billing extension in an integrated revenue or invoice flow through Oracle Projects to verify the logic that you just added. Continue this cycle for all of your SQL statements to be included in your billing extension procedure.

You may take another approach by adding all of your logic to the billing extension and then performing integrated testing. This method is harder to debug when you encounter problems.

Step 4 Do full integrated testing of billing extension

After your billing extension logic is complete, you need to perform full integrated testing to validate all of the business cases and conditions

that your billing extension must handle. This is where you use the business cases and test plans that you created in the design stage of the your billing extension implementation.

You must ensure that your billing extension works when using both project level and task level funding, if your company uses both levels of funding.

If you have written a transaction dependent billing extension, you should test the processing flow for these adjustment actions to ensure that your billing extension properly processes transactions with these adjustment actions:

- Revenue recalculation with and without change in the amount
- Transfer to the same project, which results in the same amount
- Transfer to a different task, which results in a different amount
- Split transaction
- Transfer to a different project
- Billable to non–billable reclassification

Once you have verified all of the integrated test cases, you have completed your billing extension implementation.

Other Debugging Tips

- Make sure that the name seeded in pa_billing_ extensions.procedure_name is exactly the same as the package.procedure_name if your procedure is stored in the database
- Make sure that the package.procedure_name does not exceed 30 characters
- Make sure that your procedure is compiled and stored in the database
- Make sure that there is not another invalid or outdated procedure executing instead of the procedure you intend to execute. Inactivate all other extensions at the appropriate level to ensure that only the extension you expect to execute is executing.

Defining Billing Extensions

You define billing extensions to automatically calculate and create revenue and invoice amounts.

When you define billing extensions, you specify detailed information that determines when the billing extensions are called, which processes call them, and what information is required upon entry of the billing extension.

Some extensions are provided by Oracle Projects. These extensions are all marked with a checkmark in the Predefined flag check box. When this box is checked, it is not possible to change the contents of the following fields:

- Procedure
- Order
- Revenue Budget Type
- Calling Processes
- Required Inputs
- Other Parameters
- Calling Place

See Also

Overview of Client Extensions: page 7 – 2

Fremont Corporation's Billing Extension for Communication Surcharge

Fremont Corporation defines one billing extension for communication surcharge. This billing extension calculates communication charge as a percent of the amount invoiced.

Billing Extension Name: Communicatin Charge

Calling Process: Invoice

Default Event Values:

- **Event Type**: Surcharge
- **Event Description:** Communication Charge

- **Check Boxes Checked:**
  - Adjustment Processing
  - Regular Processing
  - Percentage
  - Project Specific

Default Budget Types:

- **Cost:** Approved Cost Budget
- **Revenue:** Approved Revenue Budget

Case Study: Surcharge

This case study demonstrates how to use a billing extension to add surcharges to project invoices.

Business Rule

The first step in the design process is to determine the business rule that you want to solve using client extensions.

Business Rule: Surcharge

Charge an additional surcharge to an invoice based on a percentage of the labor amount invoiced. This surcharge is referred to as *Communication Charge.*

Business Requirements

After you define the business rule you want to solve using client extensions, list the business requirements behind the business problem. This will help ensure that you are acknowledging all of the aspects of the business problem during the design stage.

- The surcharge is applicable only for projects for which it is negotiated. Project users specify the communication charge when they record the billing terms during project setup.
- You calculate this surcharge as follows:
  - *Surcharge = Surcharge Percentage × Labor Amount Invoiced*
- Usually, the percentage is 2%. However, some project managers are beginning to negotiate 2.5% or 3% surcharges.

Required Extensions

You have determined that you will create a **billing extension** to automatically handle the Communication Charge within the invoicing cycle.

Suggestion: To review the sample PL/SQL code that corresponds to the implementation of this case study, view the file PAXITMPS.pls.

Additional Implementation Data

You must define additional data for this billing extension which includes the following:

- Event type of *Surcharge* with an event type classification of Automatic
- Descriptive flexfield segment on the Communication Charge billing assignments to hold the event description that users can enter to override the default description
- Descriptive flexfield segment on the Communication Charge billing extension to hold the corporate default percentage for communication charge

In addition, you must include the steps to enter a communication charge for projects in your company's procedures manual.

Design Requirements

You must consider and answer these additional questions for your billing extension.

Revenue or Invoice Amount?

Are you calculating a revenue amount, an invoice amount, or both? Are the amounts generated during revenue accrual, invoice generation, or both?

- The Communication Surcharge generates only an invoice amount during the invoice generation process. There is no effect on revenue.

How is the Amount Calculated?

What are the inputs to the calculation?

Surcharge = Surcharge Percentage X Labor Invoiced

What is the Calling Place?

This billing extension is called in both Regular and Adjustment processing, to account for regular transactions and for revenue and invoice credits.

How are the Inputs Derived?

- Surcharge Percentage is entered by a project user who defines the billing terms of the project. This will be entered on the billing assignment. If the percent is not specified, read the corporate default from the descriptive flexfield.

- Labor Amount Invoiced is the labor bill amount on an invoice, excluding overtime billed on the invoice.

How is the Amount Processed?

You need to determine how the amounts are processed for different purposes: 1) for reporting purposes (2) for accounting purposes, (3) for invoicing?

- There are no special reporting requirements
- There is no special accounting effect for an invoicing event.
- The default event description for the billing extension is *Communication Charge*. The project users can override the value by setting the optional descriptive flexfield segment called 'Event Description', which will be used to override the default event description.

Automatic Event Attributes?

You need to determine the various attributes of the automatic event, including: event type, event organization, event description, completion date.

- The event uses the default event type of *Surcharge* from the billing extension definition.
- The event organization is defaulted to the project or task organization. This organization is not used in processing or reporting these events.
- The event description is set as noted in the previous question.
- The completion date is set to the bill through date of the invoice.

When is the Surcharge Billing Extension Used?

Under what conditions is this calculation used? What types of projects? What types of billing terms?

- The communication surcharge is applicable for all projects for which it is negotiated.

How is the Billing Extension Processed for Adjustments?

Adjustments are defined as revenue credits or invoice credit memos, based on other transactions.

- The surcharge must be accounted for on all invoices and invoice credit memos.

Can This Billing Extension be Called with other Billing Extensions?

Can this billing extension be called with other billing extensions on the same project / task? If so, what is the dependency and order of your billing extensions?

- A project can have a communication surcharge along with other billing extensions. The communication surcharge must be processed before the other billing extensions.

What is the Processing if Some Input Values Cannot be Found?

- If no percentage is specified on the billing assignment, use the corporate default value of 2%. This default value is held on the billing extension definition in a descriptive flexfield.
- If no labor is billed, then no surcharge is billed.

How is the Logic Affected if the Inputs Change?

- The surcharge percentage could change, but the user must disable the existing billing assignment and enter a new billing assignment with a new percentage. This new percentage is then automatically processed.

Is there a Limit on the Amount Calculated?

Is there a limit on the amount calculated? If so, what is the logic?

- There is no specific limit on the communication charge.

Funding Level?

Are there implications of the level at which the project is funded – either the project level or the top task level? If so, what?

- There are no special implications.

See Also

Designing Client Extensions: page 7 – 6

Billing Extension Definition

With the answers from these questions and your understanding of the billing extension definition, you can specify the billing extension definition of Communication Charge. An example is shown below.

> **Note:** The Percentage is not a required input for every billing assignment of Communication Charge, since there is a corporate default percentage that will be used when project users do not enter a negotiated percentage.

> **Suggestion:** You can use the same PL/SQL procedure for another billing extension that uses the same logic of adding a surcharge based on a percentage multiplied by the labor amount invoiced.

Example Billing Extension:

Billing Extension Name: Communicatin Charge

Procedure: pa_demo_surcharge.execute

Description: Calculate surcharge to invoice based on percentage of labor invoiced

Order: 20

Default Event Values:

- **Event Type**: Surcharge
- **Event Description:** Communication Charge

Calling Place:

- **Revenue:** No
- **Invoice:** Yes

Calling Place:

- **Preprocessing:** No
- **Adjustment:** Yes
- **Regular:** Yes
- **Post–Processing:** No

Required Inputs:

- **Amount:** No
- **Percentage:** No

Other Parameters:

- **Product–Specific:** Yes
- **Transaction Independent:** No

Testing

You specify the following test cases to use in testing your billing extension procedure.

Scenario	Run	Inv Num	Inv Num Credited	Inv Amt	Invoice Labor Amt	%	Comm Charge Amt
No labor invoiced	1	1		1000	0	2	0
Credit memo	2	2	1	–500	0	2	0
Labor invoiced for first time		3		12000	10000	2	200
Credit memo due to rate change	3	4	3	–5000	–5000	2	–100
Labor with new bill rates		5		6000	6000	2	120
Communication Charge % was changed	4	6		5000	5000	3.5	175
Totals				18500	16000		395

Table 10 – 18 Example Test Cases for Communication Charge (Page 1 of 1)

You now have all of the components of your functional design to give to your technical resource for writing the PL/SQL procedures.

Cost Accrual Billing Extension

You can use the cost accrual billing extension client extension to apply your company's business rules to your cost accrual procedures.

Description

The extension includes the following items:

Item	Name
Body template	PAXICOSB.pls
Specification template	PAXICOSS.pls
Package	PA_REV_CA

Table 10 – 19 Cost Accrual Billing Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

For more information about using the Cost Accrual Extension, see Revenue–Based Cost Accrual, *Oracle Project Billing User Guide*

Calculation Procedure (calc_ca_amt)

The calculation procedure (calc_ca_amt) is the main procedure for calculating and generating the cost accrual entries. The parameters for this procedure are shown in the following table:

Parameter	Type	Usage	Description
x_project_id	NUMBER	IN	Project identifier
x_top_task_id	NUMBER	IN	Identifier of the top task
x_calling_place	VARCHAR2	IN	This parameter specifies where the billing extension is called in the revenue or invoice program. Possible values are PRE, POST, REG, or ADJ
x_amount	NUMBER	IN	Amount of the transaction
x_percentage	NUMBER	IN	Cost accrual percentage

Table 10 – 20 Parameters for cost accrual calculation procedure

Parameter	Type	Usage	Description
x_rev_or_bill_date	DATE	IN	The accrue through date if called by revenue generation, or the bill through date if called by invoice generation.
x_billing_assignment_id	NUMBER	IN	Identifier of the billing assignment associated with the transaction
x_billing_extension_id	NUMBER	IN	Identifier of the billing extension being processed. Use this to select information (such as descriptive flexfield values) from the billing extension definition.
x_request_id	NUMBER	IN	Request ID of the current run

Table 10 – 20 Parameters for cost accrual calculation procedure

PSI Cost Accrual Procedure (get_psi_cols)

The PSI cost accrual procedure (get_psi_cols) displays the cost accrual columns in Project Status Inquiry. The parameters for this procedure are shown in the following table:

Parameter	Type	Usage	Description
x_project_id	NUMBER	IN	Project identifier
x_top_task_id	NUMBER	IN	Identifier of the top task
x_resource_list_member_id	NUMBER	IN	Identifier of the resource list member
x_cost_budget_type_code	VARCHAR2	IN	Cost budget type code
x_rev_budget_type_code	VARCHAR2	IN	Revenue budget type code
x_status_view	VARCHAR2	IN	Identifier of the status folder: PROJECTS, TASKS, or RESOURCES
x_pa_install	VARCHAR2	IN	The identifier of the Oracle Projects product installed: BILLING or COSTING. BILLING includes all default PSI columns. COSTING includes all but the actual revenue and revenue budget columns.
x_derived_col_1 through x_derived_col3	VARCHAR2	OUT	The alphanumeric derived columns. Each can have up to 255 characters.
x_derived_col_4 through x_derived_col_33	NUMBER	OUT	The numeric derived columns.

Table 10 – 21 Parameters for PSI cost accrual procedure

Parameter	Type	Usage	Description
p_revenue_ptd	NUMBER	IN	Percentage for accruing period–to–date revenue
p_revenue_itd	NUMBER	IN	Percentage for accruing inception–to–date revenue

Table 10 – 21 Parameters for PSI cost accrual procedure

Verify Project Status for Cost Accrual Procedure (verify_project_status_ca)

This procedure is called when a user changes a project's status. The parameters are shown in the following table:

Parameter	Type	Usage	Description
x_calling_module	VARCHAR2	IN	Module from which the extension is called.
x_project_id	NUMBER	IN	Identifier of the project
x_old_proj_status_code	VARCHAR2	IN	Existing status code for the project
x_new_proj_status_code	VARCHAR2	IN	New status code for the project
x_project_type	VARCHAR2	IN	Project type of the project
x_project_start_date	DATE	IN	Start date of the project
x_project_end_date	DATE	IN	End date of the project
x_public_sector_flag	VARCHAR2	IN	Public sector indicator
x_attribute_category	VARCHAR2	IN	Descriptive flexfield context
x_attribute1 through x_attribute10	VARCHAR2	IN	Descriptive flexfield segments
x_pm_product_code	VARCHAR2	IN	The project management product code
x_err_code	NUMBER	OUT	The error handling code
x_warnings_only_flag	VARCHAR2	OUT	Indicates if the procedure had only warning messages

Table 10 – 22 Parameters for verify project status procedure

Check Cost Accrual Procedure (check_if_cost_accrual)

This procedure checks whether a project has cost accrual, and sets the variables from attribute columns 11 through 15 of the billing extension. The parameters are shown in the following table:

Parameter	Type	Usage	Description
p_project_id	NUMBER	IN	The identifier of the project
x_cost_accrual_flag	VARCHAR2	IN OUT	Indicates if the project has cost accrual
x_funding_flag	VARCHAR2	IN OUT	Indicates whether the project has funding
x_ca_event_type	VARCHAR2	IN OUT	Cost accrual event type
x_ca_contra_event_type	VARCHAR2	IN OUT	Cost accrual contra event type
x_ca_wip_event_type	VARCHAR2	IN OUT	Cost accrual WIP event type
x_ca_budget_type	VARCHAR2	IN OUT	Cost accrual budget type

Table 10 – 23 Parameters for check cost accrual procedure

Cost Accrual Identification Extension

Use this extension to identify cross charged projects that use cost accrual during revenue generation. See: Revenue–Based Cost Accrual, *Oracle Project Billing User Guide* and Generate Draft Revenue, *Oracle Projects Fundamentals*.

Description

The extension includes the following items:

Item	Name
Body template	PAICPCAB.pls
Specification template	PAICPCAS.pls
Package	PA_CC_CA
Procedure	identify_ca_projects

Table 10 – 24 Cost Accrual Identification Extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Parameters

The extension uses the following parameters:

Parameter	Type	Usage	Description
p_project_id	NUMBER	IN	Project identifier
x_cost_accrual_flag	VARCHAR2	OUT	Flag identifying cost accrual projects. Value is Y or N.

Table 10 – 25 Parameters for Cost Accrual Identification Extension

Labor Billing Extensions

Labor billing extensions allow you to derive labor billing amounts for individual labor transactions. You can use labor billing extensions to implement unique labor billing methods. Some examples of labor billing extensions you may define are:

- Bill overtime premium hours at cost
- Bill based on volume of work performed

Processing

Oracle Projects processes labor billing extensions for activity based billing during revenue generation. During processing, if Oracle Projects encounters a transaction that has a derived bill amount from a labor billing transaction, it skips the standard bill amount and rate calculation section of the revenue process for that transaction.

See Also

Revenue Flow, *Oracle Project Billing User Guide*

Designing Labor Billing Extensions

Consider the following design issues for labor billing extensions:

- What are the conditions and circumstances in which you cannot use the standard, activity based billing methods (identified by the WORK distribution rule) supported by Oracle Projects?
- How is the bill amount calculated in these cases?
- How do you identify labor transactions that meet these conditions?
- How do you store rates and other information that your calculations may require? How are the rates and other information maintained?

- What are the exception conditions for your labor billing extension? What is the exception handling if you cannot find a rate that should exist?

Writing Labor Billing Extensions

The Labor Billing Extensions is called during the revenue generation process to determine labor revenue and billing amounts.

The extension is identified by the following items:

Item	Name
Body template	PAXICTMB.pls
Specification template	PAXICTMS.pls
Package	PA_Client_Extn_Billing
Procedure	Calc_Bill_Amount

Table 10 – 26

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Package.Procedure

PA_Client_Extn_Billing.Calc_Bill_Amount

The following table lists the parameters that Oracle Projects provides for the labor billing extension.

Parameter	Usage	Type	Description
x_transaction_type	IN	VARCHAR2	An identifier that distinguishes between actual and forecast transactions. This enables you to process forecast and actual transactions differently. The default value is ACTUAL. Forecast transactions can have the following transaction types: ROLE ASSIGNMENT
x_expenditure_item_id	IN	NUMBER	The identifier of the expenditure item.
x_sys_linkage_function	IN	VARCHAR2	The expenditure type class of the expenditure item.
x_amount	IN OUT	NUMBER	The bill amount in billing transaction currency
x_bill_rate_flag	IN OUT	VARCHAR2	Indicates if bill rate should be set.
x_status	IN OUT	NUMBER	The status of the procedure.
x_bill_trans_currency_code	OUT	VARCHAR2	Identifier of the billing transaction currency for an expenditure item. If the value is null, the project functional currency is used.
x_bill_txn_bill_rate	OUT	NUMBER	Bill rate in the billing transaction currency
x_markup_percentage	OUT	NUMBER	Markup percentage (if markup is used to derive the bill amount)
x_rate_source_id	OUT	NUMBER	Identifies the rate source from which the rate was derived. This is for audit purposes only.

Table 10 – 27 (Page 1 of 1) Labor Billing Extension Parameters

Additional Information About Parameters

Using Bill Rate

Return one of the following values as the x_bill_rate_flag parameter value to specify if the amount that you have derived is based on a bill rate or a percent markup:

- B (specifies bill rate)

- null or value other than B (specifies markup)

If you specify that your amount is based on a bill rate, Oracle Projects populates the bill rate of the expenditure item by dividing the bill amount by the number of hours. If you specify that your amount is a markup, Oracle Projects does not set the bill rate.

Using Status

Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

x_status = 0	The extension executed successfully.
x_status < 0	An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file and rolls back the transactions processed for the entire project.
x_status > 0	An application error occurred. Oracle Projects writes a rejection reason to PA_EXPENDITURE_ITEMS.REV_DIST_ REJECTION_CODE and does not mark items as revenue distributed. You can review the rejection reason in the revenue generation exception report.

Retention Billing Extension

Use this extension define your company's business rules to bill withheld amounts. If you use this extension, the invoice generation process selects projects that have met the conditions defined in the extension and have a net retention balance that has not been billed.

See Also

Retention Billing, *Oracle Project Billing User Guide*

Writing Retention Billing Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXBRTCB.pls
Specification template	PAXBRTCS.pls
Package	pa_client_extn_retention
Procedure	BILL_RETENTION

Table 10 – 28

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

The following table lists the parameters that Oracle Projects provides for retention billing extension.

Parameter	Usage	Type	Description
P_Customer_ID	IN	NUMBER	Identifier of the customer for whom retention is to be billed
P_Project_ID	IN	NUMBER	Identifier of the project for which retention is to be billed. This parameter is required.
P_Top_Task_ID	IN	NUMBER	Identifier of the top task for which retention is to be billed. Value is passed only if the retention level is top task.
X_Bill_Retention_Flag	OUT	VARCHAR2	A value of 'Y' indicates that the retention will be billed using the percentage or amount specified in the extension.
X_Bill_Percentage	OUT	NUMBER	Retention billing percentage
X_Bill_Amount	OUT	NUMBER	Retention bill amount

Table 10 – 29 (Page 1 of 1) Retention Billing Extension Parameters

Automatic Invoice Approve/Release Extension

The Automatic Invoice Approve / Release Extension allows you to make automatic approval and release of invoices a part of the Generate Draft Invoice process.

Processing

Oracle Projects calls the Automatic Invoice Approve / Release Extension during invoice generation. During processing, if the extension returns an approval flag or release flag set to *yes*, then the process approves (and releases, if applicable) the invoice.

Designing Invoice Approve/Release Extensions

You must determine to what extent the Automatic Invoice Approve / Release Extension will be used across your projects. We recommend that you consider these design issues:

- What are the conditions and circumstances that require your project invoices to be automatically approved?
- What are the conditions and circumstances that require your project invoices to be automatically approved and released?
- What types of projects need to have this feature implemented?

Writing Automatic Invoice Approve/Release Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXPIACB.pls
Specification template	PAXPIACS.pls
Package	pa_client_extn_inv_actions

Table 10 – 30

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter

types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

The names of the procedures are:

- approve_invoice
- release_invoice

Package Procedures

pa_client_extn_inv_actions.approve_invoice

The following table lists the parameters that Oracle Projects provides for the invoice approval extension.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project to which the draft invoice number is attached.
p_draft_invoice_num	IN	NUMBER	The draft invoice number.
p_invoice_class	IN	VARCHAR2	The class of the invoice.
p_project_amount	IN	NUMBER	Amount of the invoice in the project currency.
p_project_currency_code	IN	VARCHAR2	The project currency code.
p_inv_currency_code	IN	VARCHAR2	The invoice currency code.
p_invoice_amount	IN	NUMBER	Amount of the invoice in the invoice currency.

Table 10 – 31 (Page 1 of 2) Automatic Invoice Approve Extension Parameters

Parameter	Usage	Type	Description
x_approve_flag	OUT	VARCHAR2	Invoice approval flag. Valid values: Y = Yes (approve invoice) any other value = do not approve
x_status	OUT	NUMBER	Status of the procedure: 0 = successful execution <0 = Oracle error >0 = application error

Table 10 – 31 (Page 2 of 2) Automatic Invoice Approve Extension Parameters

pa_client_extn_inv_actions.release_invoice

The following table lists the parameters that Oracle Projects provides for the invoice release extension.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project to which the draft invoice number is attached.
p_draft_ invoice_num	IN	NUMBER	The draft invoice number.
p_invoice_class	IN	VARCHAR2	The class of the invoice.
p_project_amount	IN	NUMBER	Amount of the invoice in the project currency.
p_project_currency_ code	IN	VARCHAR2	The project currency code.
p_inv_currency_ code	IN	VARCHAR2	The invoice currency code.
p_invoice_amount	IN	NUMBER	Amount of the invoice in the invoice currency.

Table 10 – 32 (Page 1 of 2) Invoice Release Extension Parameters

Parameter	Usage	Type	Description
x_release_flag	OUT	VARCHAR2	Invoice release flag. Valid values: Y = Yes (release invoice) any other value = do not release
x_ra_invoice_date	OUT	DATE	Receivable invoice date. Validation on this parameter is performed only when x_release_flag = Y.
x_ra_invoice_num	OUT	NUMBER	Receivable invoice number. If automatic invoice numbering is active, then this parameter is not required. Validation on this parameter is performed only when x_release_flag = Y.
x_status	OUT	NUMBER	Status of the procedure: 0 = successful execution <0 = Oracle8 error >0 = application error

Table 10 – 32 (Page 2 of 2) Invoice Release Extension Parameters

Additional Information About Parameters

Using Invoice Class

The valid values of x_invoice_class are:

INVOICE regular invoice

CREDIT_MEMO crediting invoice

WRITE_OFF write–off invoice

CANCEL canceling invoice

Using Status

Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

x_status = 0	The extension executed successfully.
x_status < 0	An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.
x_status > 0	An application error occurred. Oracle Projects writes a rejection reason to the PA_DISTRIBUTION_WARNINGS table. The invoice is not approved or released.

Output Tax Extension

In the Tax Defaults implementation option, you set up a hierarchy for determining default tax codes for invoice lines. One of the sources the system can use to find default tax codes is the Output Tax client extension.

Oracle Projects calls the Output Tax extension during the Generate Draft Invoices process, if it has not yet found the output tax code using the Tax Defaults hierarchy. You can modify the extension to satisfy your business rules for assigning the default output tax code for invoice lines.

Description

The extension is identified by the following items:

Item	Name
Specification template	PAXPOTXS.pls
Body template	PAXPOTXB.pls
Package	pa_client_extn_output_tax
Procedure	get_tax_codes

Table 10 – 33 Output Tax extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Get_Tax_Codes Procedure

The get_tax_codes procedure assigns a tax code to an invoice line. This procedure uses the following parameters:

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	The identifier of the project
p_customer_id	IN	NUMBER	The identifier of the customer

Table 10 – 34 get_tax_codes parameters (Page 1 of 2)

Parameter	Usage	Type	Description
p_bill_to_site_use_id	IN	NUMBER	The bill–to site
p_ship_to_site_use_id	IN	NUMBER	The ship–to site
p_set_of_books_id	IN	NUMBER	The set of books associated with the project
p_expenditure_item_id	IN	NUMBER	Identifier of the expenditure item
p_event_id	IN	NUMBER	The identifier of the event
p_line_type	IN	VARCHAR2	The type of invoice line (Event, Expenditure, or Retention)
p_request_id	IN	NUMBER	The request ID of the Generate Draft Invoices process
p_user_id	IN	NUMBER	The identifier of the user who ran the Generate Draft Invoices process
x_vat_tax_id	OUT	NUMBER	The output tax code

Table 10 – 34 get_tax_codes parameters (Page 2 of 2)

See Also

Setting Up Invoice Line Tax Codes, *Oracle Projects Implementation Guide*

Tax Defaults, *Oracle Projects Implementation Guide*

Receivables Installation Override

The Receivables Installation Override client extension allows you to use a third–party receivables system for the majority of your receivables functionality, yet have the ability to import customer data from Oracle Receivables. Without this client extension, you can only import customer data with a full installation of Oracle Receivables.

To use this capability, you must complete a *full* installation of Oracle Receivables, then override the installation mode to *shared*, using the Receivables Installation Override extension.

Warning: Do not override a shared Receivables installation to full installation mode. This client extension is only intended for overriding a full installation to shared mode.

The following conditions exist when you override the installation to shared mode:

- The Tax Code fields are disabled in all windows where they appear.
- The GL date for receivables invoices is calculated based on GL periods, rather than Oracle Receivables periods.

If you override the Receivables installation, you can use function security to disable functions that are not available with a standard shared Receivables installation, such as *Invoice: AR Invoice* (drill down to Oracle Receivables to view an invoice).

Warning: You must disable the Invoice: Write–Off function, as attempting to create write offs will cause processing problems.

This extension is called by the Interface Invoices to Receivables process.

Description

The extension is identified by the following items:

Item	Name
Specification template	PAPARICS.pls
Body template	PAPARICB.pls

Table 10 – 35 Receivables installation override extension

Item	Name
Package	pa_override_ar_inst
Procedure	get_installation_mode

Table 10 – 35 Receivables installation override extension

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL/SQL Procedures: page 7 – 8.

Get_Installation_Mode Procedure

The get_installation_mode procedure returns an installation mode to the calling program.

This procedure uses the following parameters:

Parameter	Usage	Type	Description
p_ar_inst_mode	IN	VARCHAR2	The input mode (mode in which Oracle Receivables is installed)
x_ar_inst_mode	OUT	VARCHAR2	The output (override) installation mode

Table 10 – 36 get_installation_mode parameters (Page 1 of 1)

Modifying the Get_Installation_Mode Procedure

The default procedure includes the following PL/SQL statement:

```
x_ar_inst_mode := p_ar_inst_mode
```

To override your full installation of Oracle Receivables to a shared mode, replace the statement above with the following statement:

```
x_ar_inst_mode := 'S'
```

AR Transaction Type Extension

The AR Transaction Type Extension enables you to determine the AR transaction type when you interface invoices to Oracle Receivables.

Processing

Oracle Projects calls the AR Transaction Type Extension during the Transfer Invoices to Oracle Receivables process.

Writing AR Transaction Type Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXPTRXB.pls
Specification template	PAXPTRXS.pls
Package	pa_client_extn_inv_transfer
Procedure	get_ar_trx_type

Table 10 – 37

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Package Procedure

pa_client_extn_inv_transfer.get_ar_trx_type

The following table lists the parameters that Oracle Projects provides for the AR transaction type extension.

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project to which the draft invoice number is attached.
p_draft_ invoice_num	IN	NUMBER	The draft invoice number.
p_invoice_class	IN	VARCHAR2	The class of the invoice.
p_project_amount	IN	NUMBER	Amount of the invoice in the project currency.
p_project_currency_ code	IN	VARCHAR2	The project currency code.
p_inv_currency_ code	IN	VARCHAR2	The invoice currency code.
p_invoice_amount	IN	NUMBER	Amount on the invoice in the invoice currency.
p_ar_trx_type_id	IN	NUMBER	Identifier of the AR Transaction Type to be used for the invoice. Oracle Projects uses its setup tables to determine the default AR transaction type and then passes it to the template.
x_ar_trx_type_id	OUT	NUMBER	Identifier of the AR Transaction Type determined by the extension. After validation, Oracle Projects uses this transaction type to interface invoices to Oracle Receivables.
x_status	OUT	NUMBER	Status of the procedure: 0 = successful execution <0 = Oracle error >0 = application error

Table 10 – 38 (Page 1 of 1) AR Transaction Type Parameters

Additional Information About Parameters

Using Invoice Class

The valid values of x_invoice_class are:

INVOICE	regular invoice
CREDIT_MEMO	crediting invoice
WRITE_OFF	write–off invoice
CANCEL	canceling invoice

Using Status

Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

x_status = 0	The extension executed successfully.
x_status < 0	An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.
x_status > 0	An application error occurred. Oracle Projects writes a rejection reason to the PA_DISTRIBUTION_WARNINGS table. The invoice is not approved or released.

CHAPTER

11

Oracle Project Resource Management Client Extensions

This chapter describes the client extensions in the Oracle Project Resource Management application.

Assignment Approval Changes Extension

This client extension enforces the following conditions to determine whether an approval is required for an assignment:

- Change in duration

  A change in the dates of an assignment requires approval, because it affects the schedule and availability of the resource.

- Change in work type

  A change in the work type on an assignment can affect the billability and utilization percentage of the resource and therefore requires approval.

Processing

The default project assignment approval workflow process calls the assignment approval changes extension.

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PARAAPCB.pls
Specification template	PARAAPCS.pls
Package	pa_client_extn_asgnmt_apprvl

Table 11 – 1

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

Procedures and Functions

The function in the assignment approval changes extension is described in this section.

Changed Approval Items Function

The function name is: **is_asgmt_appr_items_changed**.

This function returns a VARCHAR2 value (either Y or N) to indicate whether approval items have been changed.

The following table lists the parameter that is used by the is_asgmt_appr_items_changed function.

Parameter	Usage	Type	Description
p_assignment_id	IN	NUMBER	The identifier of the assignment

Table 11 – 2 (Page 1 of 1) Changed Approval Items Parameter

Assignment Approval Notification Extension

You can use this client extension to customize the list of default contacts (recipients) used by the assignment approval workflow.

Processing

The default project assignment approval workflow process calls the assignment approval notification extension.

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PARAWFCB.pls
Specification template	PARAWFCS.pls
Package	pa_client_extn_asgmt_wf

Table 11 – 3

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

User List Parameters

The USERS_LIST_TBLTYP parameters for this package are listed in the following table:

Parameter Name	Data Type	Description	Mandatory
USER_NAME	VARCHAR2	The workflow user name of the approver	Yes
PERSON_ID	NUMBER	The person identifier of the approver	Yes

Parameter Name	Data Type	Description	Mandatory
TYPE	VARCHAR2	The type of user, such as RESOURCE_MANAGER or PRIMARY_CONTACT	Yes
ROUTING_ORDER	NUMBER	The order in which the approvals should be submitted	No. For FYI notification recipients, this value is ignored since such notifications are sent to all recipients at the same time

Table 11 – 4 Users List Parameters (Page 2 of 2)

Procedures

The procedures in the assignment approval notification extension are described in this section.

Generate Assignment Approvers

The procedure name is: **generate_assignment_approvers**.

This procedure generates a list of approvers for the assignment. Oracle Projects sends the list of default approvers to this procedure. The procedure then makes user–requested changes and provides a modified list accordingly.

Approvers added through this process are not visible on the Assignment Approver page. However, users can see the name of the current approver on the Assignment Details page.

The following table lists the parameters that are used by the generate assignment approvers procedure.

Parameter Name	Type	Data Type	Description
P_ASSIGNMENT_ID	IN	NUMBER	The unique identifier of the assignment
P_PROJECT_ID	IN	NUMBER	The unique identifier of the project
P_IN_LIST_OF_APPROVERS	IN	References the USERS_LIST_TBLTYP	Input list of notification recipients
X_OUT_LIST_OF_APPROVERS	OUT	References the USERS_LIST_TBLTYP	Output list of notification recipients
X_NUMBER_OF_APPROVERS	OUT	NUMBER	Number of recipients

Table 11 – 5 Generate Assignment Approvers Parameters (Page 1 of 1)

Generate Notification Recipients

The procedure name is: **generate_nf_recipients**.

This procedure generates a list of recipients for notifications. Oracle Projects sends the list of default approvers to this procedure. The procedure makes user–requested changes and returns a modified list.

This procedure is used by the following FYI notifications:

- Assignment Approval Notification
- Assignment Rejection Notification
- Assignment Cancellation Notification

The following table lists the parameters that are used by the generate notification recipients procedure.

Parameter Name	Type	Data Type	Description
P_ASSIGNMENT_ID	IN	NUMBER	The unique identifier of the assignment
P_PROJECT_ID	IN	NUMBER	The unique identifier of the project

Table 11 – 6 Generate Notification Recipients Parameters (Page 1 of 2)

Parameter Name	Type	Data Type	Description
P_NOTIFICATION_TYPE	IN	VARCHAR2	Type of notification. Valid values are: APPROVAL_FYI and REJECTION_FYI
P_IN_LIST_OF_RECIPIENTS	IN	References the USERS_LIST_TBLTYP	Input list of notification recipients
X_OUT_LIST _OF_RECIPIENTS	OUT	References the USERS_LIST_TBLTYP	Output list of notification recipients
X_NUMBER_OF_RECIPIENTS	OUT	NUMBER	Number of recipients

Table 11 – 6 Generate Notification Recipients Parameters (Page 2 of 2)

Set Timeout and Reminders

The procedure name is: **set_timeout_and_reminders**.

This procedure provides the reminder parameters, such as the waiting period between reminders and the number of reminders that are issued before the workflow process is canceled.

The following table lists the parameters that are used by the generate notification recipients procedure.

Parameter Name	Type	Data Type	Description
P_ASSIGNMENT_ID	IN	NUMBER	The unique identifier of the assignment
P_PROJECT_ID	IN	NUMBER	The unique identifier of the project
X_WAITING_TIMES	OUT	NUMBER	The maximum amount of time to wait before sending a reminder
X_NUMBER_OF_REMINDERS	OUT	NUMBER	The maximum number of reminders to send before aborting the process

Table 11 – 7 Set Timeout and Reminders Parameters (Page 1 of 1)

Candidate Notification Workflow Extension

You can use this client extension to customize the candidate workflow processes.

Processing

The default New Candidate and Candidate Assigned workflow processes call the candidate notification workflow extension.

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PARCWFCB.pls
Specification template	PARCWFCS.pls
Package	pa_client_extn_cand_wf

Table 11 – 8

Attention: Do not change the name of the extension procedures or parameters. Also, do not change the parameter types or parameter order in your procedure. After you write a procedure, compile it and store it in the database. For more information, see: Writing PL / SQL Procedures: page 7 – 8.

The USERS_LIST_TBLTYP parameters for this package are shown in the following table:

Parameter Name	Required	Data Type	Description
USER_NAME	Yes	VARCHAR2	The workflow user name of the approver
PERSON_ID	Yes	NUMBER	The person ID of the approver

Table 11 – 9 Candidate Notification Workflow Extension Parameters (Page 1 of 2)

Parameter Name	Required	Data Type	Description
TYPE	Yes	VARCHAR2	The type of user, such as RESOURCE_MANAGER or PRIMARY_CONTACT
ROUTING_ORDER	No	NUMBER	The order in which the approvals should be submitted. (For FYI notification recipients, this value is ignored since such notifications are sent to all recipients at the same time.)

Table 11 – 9 Candidate Notification Workflow Extension Parameters (Page 2 of 2)

Generate Notification Recipients

This package contains the procedure **generate_nf_recipients**.

This procedure generates a list of recipients for the various notifications. Oracle Projects sends the list of default approvers to this procedure. The procedure makes changes and provides a modified list. This procedure is used by the FYI notification Candidate Nominated Notification.

The following table lists the parameters that are used by this procedure.

Parameter Name	Type	Data Type	Description
P_PROJECT_ID	IN	NUMBER	The unique identifier of the project
P_ASSIGNMENT_ID	IN	NUMBER	The unique identifier of the assignment
P_CANDIDATE_NUMBER	IN	NUMBER	The unique identifier of the candidate
P_NOTIFICATION_TYPE	IN	VARCHAR2	Type of notification. Valid values are: PENDING_, REVIEW_FYI, and DECLINED_FYI
P_IN_LIST_OF_RECIPIENTS	IN	References the USERS_LIST_TBLTYP	Input list of notification recipients

Table 11 – 10 Generate Notification Recipients Parameters (Page 1 of 2)

Parameter Name	Type	Data Type	Description
X_OUT_LIST _OF_RECIPIENTS	OUT	References the USERS_LIST_TBLTYP	Output list of notification recipients
X_NUMBER_OF_RECIPIENTS	OUT	NUMBER	Number of recipients

Table 11 – 10 Generate Notification Recipients Parameters (Page 2 of 2)

CHAPTER

12

Oracle Project Management Client Extensions

This chapter describes the client extensions in the Oracle Project Management application.

Workplan Workflow Extension

The workplan workflow extension enables you to customize the workflow processes for submitting, approving, and publishing a workplan.

You must determine how you want to submit, approve, and publish the workplan. See Creating and Updating Workplans, *Oracle Project Management User Guide.*

Processing

The default workplan workflow process calls the workplan workflow extension.

See Also

Designing Client Extensions: page 7 – 6

Writing Workplan Workflow Extensions

The extension is identified by the following items:

Item	Name
Body template	PAXSTWCB.pls
Specification template	PAXSTWCS.pls
Package	pa_workplan_workflow_client

Table 12 – 1

Start Workflow Procedure

The procedure name is **start_workflow**.

This procedure starts the workflow process for a workplan.

The following table lists the parameters that Oracle Projects provides for the start_workflow procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type
p_item_key	IN	VARCHAR2	The workflow item key
p_process_name	IN	VARCHAR2	Name of the workflow process
p_structure_version_id	IN	NUMBER	Identifier of the structure version
p_responsibility_id	IN	NUMBER	Identifier of the user responsibility
p_user_id	IN	NUMBER	Identifier of the user
x_msg_count	OUT	NUMBER	The number of messages being sent
x_msg_data	OUT	VARCHAR2	The content of the message
x_return_status	OUT	VARCHAR2	The return status of the message

Table 12 – 2 (Page 1 of 1) Start Workflow

Select Approver Procedure

The procedure name is: **select_approver**.

This procedure determines the approver for the workplan approval process.

The following table lists the parameters that Oracle Projects provides for the select_approver procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The item type.
p_item_key	IN	VARCHAR2	The workflow item key.

Table 12 – 3 (Page 1 of 2) Select Approver

Parameter	Usage	Type	Description
actid	IN	NUMBER	Identifier of the action.
funcmode	IN	VARCHAR2	Workflow function mode.
resultout	OUT	VARCHAR2	Process result.

Table 12 – 3 (Page 2 of 2) Select Approver

Set Notification Party Procedure

The name of this procedure is **set_notification_party**.

This procedure determines which users receive workflow notifications when a workplan is submitted, approved, rejected, or published.

The following table lists the parameters that Oracle Projects provides for the set_notification_party procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	Name of the workflow process.
p_item_key	IN	VARCHAR2	The workflow item key.
p_status_code	IN	VARCHAR2	The workplan status code.
actid	IN	NUMBER	Identifier of the action.
funcmode	IN	VARCHAR2	Workflow function mode.
resultout	OUT	VARCHAR2	Process result.

Table 12 – 4 (Page 1 of 1) Set Notification Party

Budget Calculation Extensions

Budget calculation extensions enable you to control how Oracle Projects processes budgets and forecasts. You can use budget calculation extensions to facilitate budget and forecast entry by defining your own rules for calculating budget and forecast amounts, based on the quantities and raw cost amounts that you enter.

> **Note:** You can use function security to control whether users can override calculated amounts, based on user responsibility. The functions pertaining to this feature have names that begin with *Budget: Line Source*. See: Function Security in Oracle Projects, *Oracle Projects Implementation Guide*.

Types of Calculations

You can use budget calculation extensions to calculate the following budget and forecast amounts:

Raw Cost

Oracle Projects calls the budget calculation extension for raw cost when you enter a *quantity* in a cost budget or forecast plan line. If you define rules in the budget calculation extension that return a value, then Oracle Projects displays the calculated amount in the *Raw Cost* amount field.

Examples of raw cost calculation rules that you can define are:

- Calculate raw cost for an employee based on the number of hours entered
- Calculate raw cost for vehicle usage based on the number of days entered

Burdened Cost

Oracle Projects calls the budget calculation extension for burdened cost when you enter a *quantity* or a *raw cost* amount in a cost budget or forecast plan line. If you define rules in the budget calculation extension that return a value, then Oracle Projects displays the calculated amount in the *Burdened Cost* amount field.

Examples of burdened cost calculation rules that you can define are:

- Calculate raw cost and burdened cost for an employee based on the number of hours entered

- Calculate burdened cost for computer usage charges based on the raw cost entered

Revenue

Oracle Projects calls the budget calculation extension for revenue when you enter a *quantity* in a revenue budget or forecast plan line. If you define rules in the budget calculation extension that return a value, then Oracle Projects displays the calculated amount in the *Revenue* amount field.

Examples of revenue calculation rules that you can define are:

- Calculate revenue for an employee based on a standard bill rate assigned to the task
- Calculate revenue for a job based on the number of hours entered

Designing Budget Calculation Extensions

You should determine the logic and the additional data elements your client extensions require before you write them. We recommend that you consider the following design issues for budget calculation extensions:

- What conditions should be true for a budget or forecast before it can be baselined?
- What are the conditions or circumstances under which you will derive the raw, burdened, or revenue amounts?
- How will you determine the rate to calculate the amount?
- How will you store the rates: in Oracle Projects tables or in custom tables?
- When can the derived amounts be overridden by the user?
- In what order should the calculations be executed if you have multiple rules?

See Also

Designing Client Extensions: page 7 – 6

The extension is identified by the following items:

Item	Name
Body template	PAXBCECB.pls
Specification template	PAXBCECS.pls
Package	PA_Client_Extn_Budget

Table 12 – 5

The names of the procedures are:

- calc_raw_cost
- calc_burdened_cost
- calc_revenue

Package Procedures

calc_raw_cost

The following table lists the parameters that Oracle Projects provides for the budget calculation procedure for raw cost.

Parameter	Usage	Type	Description
X_budget_version_id	IN	NUMBER	The identifier of the budget or forecast version.
X_project_id	IN	NUMBER	The identifier of the project.
X_task_id	IN	NUMBER	The identifier of the task. Set to zero if budgeting or forecasting at the project level.
X_resource_list_member_id	IN	NUMBER	The identifier of the resource list member.
X_resource_list_id	IN	NUMBER	The identifier of the resource list.
X_resource_id	IN	NUMBER	The identifier of the resource.

Table 12 – 6 (Page 1 of 2) Calculate Raw Cost Parameters

Parameter	Usage	Type	Description
X_start_date	IN	DATE	The start date of the budget or forecast plan line.
X_end_date	IN	DATE	The end date of the budget or forecast plan line.
X_period_name	IN	VARCHAR2	The effective period of the budget or forecast plan line (if any).
X_quantity	IN	NUMBER	The quantity of the budget or forecast plan line.
X_raw_cost	IN OUT	NUMBER	The raw cost of the budget or forecast plan line. Oracle Projects passes in the entered amount. An amount is then returned by the extension.
X_product_code	IN	VARCHAR2	The product code of the product where the budget or forecast plan line originated.
X_error_code	OUT	NUMBER	Error handling code.
X_error_message	OUT	VARCHAR2	User–defined error message.

Table 12 – 6 (Page 2 of 2) Calculate Raw Cost Parameters

calc_burdened_cost

The following table lists the parameters that Oracle Projects provides for the budget calculation procedure for burdened cost.

Parameter	Usage	Type	Description
X_budget_version_id	IN	NUMBER	The identifier of the budget or forecast version.
X_project_id	IN	NUMBER	The identifier of the project.
X_task_id	IN	NUMBER	The identifier of the task. Set to zero if budgeting or forecasting at the project level
X_resource_list_member_id	IN	NUMBER	The identifier of the resource list member.
X_resource_list_id	IN	NUMBER	The identifier of the resource list.
X_resource_id	IN	NUMBER	The identifier of the resource.
X_start_date	IN	DATE	The start date of the budget or forecast plan line.
X_end_date	IN	DATE	The end date of the budget or forecast plan line.
X_period_name	IN	VARCHAR2	The effective period of the budget or forecast plan line (if any).
X_quantity	IN	NUMBER	The quantity of the budget or forecast plan line.
X_raw_cost	IN	NUMBER	The raw cost of the budget or forecast plan line.
X_burdened_cost	IN OUT	NUMBER	The burdened cost of the budget or forecast plan line. Oracle Projects passes in the entered amount. An amount is then returned by the extension.

Table 12 – 7 (Page 1 of 2) Calculate Burdened Cost Parameters

Parameter	Usage	Type	Description
X_product_code	IN	VARCHAR2	The product code of the product where the budget or forecast plan line originated.
X_error_code	OUT	NUMBER	Error handling code.
X_error_message	OUT	VARCHAR2	User–defined error message.

Table 12 – 7 (Page 2 of 2) Calculate Burdened Cost Parameters

Suggestion: Use the Cost Plus API to calculate the burdened cost amount using the burdened multipliers you have defined for the project or task. See: Cost Plus API: page 4 – 15.

calc_revenue

The following table lists the parameters that Oracle Projects provides for the budget calculation procedure for revenue.

Parameter	Usage	Type	Description
X_budget_version_id	IN	NUMBER	The identifier of the budget or forecast version.
X_project_id	IN	NUMBER	The identifier of the project.
X_task_id	IN	NUMBER	The identifier of the task. Set to zero if budgeting or forecasting at the project level.
X_resource_list_member_id	IN	NUMBER	The identifier of the resource list member.
X_resource_list_id	IN	NUMBER	The identifier of the resource list.
X_resource_id	IN	NUMBER	The identifier of the resource.
X_start_date	IN	DATE	The start date of the budget or forecast plan line.
X_end_date	IN	DATE	The end date of the budget or forecast plan line.

Table 12 – 8 (Page 1 of 2) Calculate Revenue Parameters

Parameter	Usage	Type	Description
X_period_name	IN	VARCHAR2	The effective period of the budget or forecast plan line (if any).
X_quantity	IN	NUMBER	The quantity of the budget or forecast plan line.
X_revenue	IN OUT	NUMBER	The revenue of the budget or forecast plan line. Oracle Projects passes in the entered amount. An amount is then returned by the extension.
X_product_code	IN	VARCHAR2	The product code of the product where the budget or forecast plan line originated.
X_error_code	OUT	NUMBER	Error handling code.
X_error_message	OUT	VARCHAR2	User–defined error message.

Table 12 – 8 (Page 2 of 2) Calculate Revenue Parameters

Additional Information About Parameters

Error Handling

Use the x_error_code, x_error_message, p_error_code, and p_error message parameters to help resolve error conditions should your procedure fail.

The x_err_code or p_error_code parameter indicates the processing status of your procedure as follows:

Suggestion: Ensure that you are returning the status of the budget calculation procedure to the procedure that you are calling the budget calculation extension from to help resolve error conditions.

x_error_code = 0 The procedure executed successfully.

x_error_code < 0 An Oracle error occurred and the process did not complete.

x_error_code > 0 An application error occurred and the process did not complete

If x_error_code or p_error_code is set to a nonzero value in the client extension, a message such as the following is displayed:

> Calculate raw cost budget client extension error <x_error_code>: <x_error_message>.

Budget Verification Extension

The budget verification extension enables you to define rules for validating a budget or forecast when it is submitted or baselined.

You should determine your requirements for submitting and baselining budgets and forecasts. For more information on submitting and baselining budgets and forecasts, see: Using Budgeting and Forecasting and Creating Budgets and Forecasts With Budgetary Controls and Budget Integration, *Oracle Project Management User Guide*.

See Also

Designing Client Extensions: page 7 – 6

Writing the Budget Verification Extension

The extension is identified by the following items:

Item	Name
Body template	PAXBCECB.pls
Specification template	PAXBCECS.pls
Package	PA_Client_Extn_Budget
Procedure	verify_budget_rules

Table 12 – 9

Package Procedure

verify_budget_rules

You can use this procedure to build additional validations that Oracle Projects performs whenever a budget or forecast is submitted or baselined. The parameter *p_event* passes a value of either *SUBMIT* or *BASELINE* to indicate the desired status of the budget or forecast being tested.

The following table lists the parameters that Oracle Projects provides for the verify budget rules procedure.

Parameter	Usage	Type	Description
p_draft_version_id	IN	NUMBER	The identifier of the budget or forecast version.
p_mark_as_original	IN	VARCHAR2	Identifies the Mark as Original request.
p_event	IN	VARCHAR2	Identifies the requested status of the budget or forecast. Value is either *SUBMIT* or *BASELINE*.
p_project_id	IN	NUMBER	The identifier of the project.
p_budget_type_code	IN	VARCHAR2	The budget or plan type code.
p_resource_list_id	IN	NUMBER	The identifier of the resource list for the budget or forecast.
p_project_type_class_code	IN	VARCHAR2	The project type class code of the project.
p_created_by	IN	NUMBER	The identifier of the person who created the budget or forecast.
p_calling_module	IN	VARCHAR2	The module that called the extension.
p_warnings_only_flag	OUT	VARCHAR2	Indicates the level of errors the procedure generated. *Y* indicates that only warnings were generated. *N* indicates that one or more errors were generated.
p_err_msg_count	OUT	NUMBER	The number of warnings and errors that the procedure generated.
p_error_code	OUT	NUMBER	Error handling code.

Table 12 – 10 (Page 1 of 2) Verify Budget Rules Parameters

Parameter	Usage	Type	Description
p_err_stage	IN OUT	VARCHAR2	Error handling stage.
p_err_stack	IN OUT	VARCHAR2	Error handling stack.

Table 12 – 10 (Page 2 of 2) Verify Budget Rules Parameters

Budget Workflow Extension

The budget workflow extension enables you to customize the workflow processes for changing the status of a budget or forecast.

Processing

Oracle Projects calls the budget workflow process to determine whether to call Oracle Workflow to baseline a budget or forecast, and which workflow process to call.

The default budget workflow process calls the budget workflow extension to determine the budget or forecast approver.

Designing Budget Workflow Extensions

Before setting up this extension, you must determine what rules you want to apply when determining whether to call Oracle Workflow to baseline a budget or forecast, and when selecting the budget or forecast approver.

Writing Budget Workflow Extensions

The extension is identified by the following items:

Item	Name
Body template	PAWFBCEB.pls
Specification template	PAWFBCES.pls
Package	pa_client_extn_budget_wf

Table 12 – 11

Package Procedures

budget_wf_is_used

When Oracle Projects determines whether to call Oracle Workflow for a budget or forecast status change, it bases the decision on the settings of the budget type or plan type, and the project type. You can use this

procedure to override those settings and to add additional requirements.

The following table lists the parameters that Oracle Projects provides for the budget_wf_is_used procedure.

Parameter	Usage	Type	Description
p_draft_version_id	IN	NUMBER	The identifier of the budget or forecast version.
p_project_id	IN	NUMBER	Identifier of the project.
p_budget_type_code	IN	VARCHAR2	The budget or plan type code.
p_pm_product_code	IN	VARCHAR2	The project management product code.
p_result	IN OUT	VARCHAR2	Result of the procedure. Value is either *Y* or *N*.
p_err_code	IN OUT	NUMBER	Error handling code.
p_err_stage	IN OUT	VARCHAR2	Error handling stage.
p_err_stack	IN OUT	VARCHAR2	Error handling stack.

Table 12 – 12 (Page 1 of 1) Budget Workflow Is Used Parameters

start_budget_wf

This procedure starts the workflow process for budget and forecast status changes. The procedure also contains the name of the workflow process that is called. The process indicated in the default procedure is PABUDWF.

The following table lists the parameters that Oracle Projects provides for the start_budget_wf procedure.

Parameter	Usage	Type	Description
p_draft_version_id	IN	NUMBER	The identifier of the budget or forecast version.
p_project_id	IN	NUMBER	Identifier of the project.
p_budget_type_code	IN	VARCHAR2	The budget or plan type code.
p_mark_as_original	IN	VARCHAR2	Indicates whether the user has requested that the budget be marked as the original budget.
p_item_type	OUT	VARCHAR2	The workflow item type.
p_item_key	OUT	VARCHAR2	The workflow item key.
p_err_code	IN OUT	NUMBER	Error handling code.
p_err_stage	IN OUT	VARCHAR2	Error handling stage.
p_err_stack	IN OUT	VARCHAR2	Error handling stack.

Table 12 – 13 (Page 1 of 1) Start Budget Workflow Parameters

verify_budget_rules

You can use this procedure to specify budget verification rules that are applied only when Oracle Workflow is used for budget and forecast status changes. This procedure is called by the procedure **pa_budget_wf.baseline_budget.**

The following table lists the parameters that Oracle Projects provides for the verify_budget_rules procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type.
p_item_key	IN	VARCHAR2	The workflow item key.

Table 12 – 14 (Page 1 of 2) Verify Budget Rules Parameters

Parameter	Usage	Type	Description
p_project_id	IN	NUMBER	Identifier of the project.
p_budget_type_code	IN	VARCHAR2	The budget or plan type code.
p_workflow_ started_by_id	IN	NUMBER	Identifier of the person who submitted the project status change.
p_event	IN	VARCHAR2	Identifies the requested status of the budget or forecast. Value is either *SUBMIT* or *BASELINE*.
p_warnings_only_ flag	OUT	VARCHAR2	Indicates the level of errors the procedure generated. *Y* indicates that only warnings were generated. *N* indicates that one or more errors were generated.
p_err_msg_count	OUT	NUMBER	Number of warnings and errors.

Table 12 – 14 (Page 2 of 2) Verify Budget Rules Parameters

select_budget_approver

This procedure is called by Oracle Workflow to determine the budget or forecast approver. You can use this procedure to add rules for determining who can approve a budget or forecast. The default procedure returns the ID of the supervisor of the person who requested the budget or forecast status change.

The following table lists the parameters that Oracle Projects provides for the select_budget_approver procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type.
p_item_key	IN	VARCHAR2	The workflow item key.
p_project_id	IN	NUMBER	Identifier of the project.

Table 12 – 15 (Page 1 of 2) Select Budget Approver Parameters

Parameter	Usage	Type	Description
p_budget_type_code	IN	VARCHAR2	The budget or plan type code.
p_workflow_ started_by_id	IN	NUMBER	Identifier of the person who requested the budget or forecast status change.
p_budget_baseliner_ id	OUT	NUMBER	Identifier of the person selected to approve the budget or forecast status change.

Table 12 – 15 (Page 2 of 2) Select Budget Approver Parameters

Control Item Document Numbering Extension

This extension enables you to create your own logic for numbering issues and change documents when automatic numbering is enabled for a control item type.

See Also

Implementing Client Extensions: page 7 – 5

Control Item Types, *Oracle Projects Implementation Guide*

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PACINRXB.pls
Specification template	PACINRXS.pls
Package	pa_ci_number_client_extn

Table 12 – 16

Procedures

Get Next Number Procedure

The procedure name is: **get_next_number**

Use this procedure to define your numbering logic. When automatic numbering is enabled for a control item type, Oracle Projects calls this procedure each time a number is assigned to an issue or a change document.

Parameters

The following table lists the parameters that are used by the Get Next Number procedure.

Parameter	Usage	Type	Description
p_object1_pk1_value	IN	NUMBER	The project identifier.
p_object1_type	IN	VARCHAR2	The business object type. For Oracle Projects, the value must be PA_PROJECTS.
p_object2_pk1_value	IN	NUMBER	The identifier of the control item type.
p_object2_type	IN	VARCHAR2	The class code of the control item type.
p_next_number	IN OUT	VARCHAR2	The generated control item number.
x_return_status	OUT	VARCHAR2	The return status of the message.
x_msg_count	OUT	NUMBER	The number of messages.
x_msg_data	OUT	VARCHAR2	The content of the message.

Table 12 – 17 Get Next Number Procedure Parameters (Page 1 of 1)

Issue and Change Workflow Extension

This extension enables you to customize the workflow processes for submitting and approving issues and change documents.

See Also

Implementing Client Extensions: page 7 – 5

Contol Item Types, *Oracle Projects Implementation Guide*

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PACIWFCB.pls
Specification template	PACIWFCS.pls
Package	pa_control_items_wf_client

Table 12 – 18

Procedures

Start Workflow Procedure

The procedure name is: **start_workflow**

Use this procedure to start the workflow process for issue and change document approval.

Parameters

The following table lists the parameters that are used by the Start Workflow procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type.
p_process_name	IN	VARCHAR2	The workflow process name.
p_item_key	IN	NUMBER	The workflow item key.
p_ci_id	IN	NUMBER	The control item identifier.
x_msg_count	OUT	NUMBER	The number of messages sent.
x_msg_data	OUT	VARCHAR2	The content of the message.
x_return_status	OUT	VARCHAR2	The return status of the message.

Table 12 – 19 Start Workflow Procedure Parameters (Page 1 of 1)

Set Control Item Approver Procedure

The procedure name is: **set_ci_approver**

Use this procedure to specify persons that can approve issues and change documents.

Parameters

The following table lists the parameters that are used by the Set Control Item Approver procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type.
p_item_key	IN	VARCHAR2	The workflow item key.
actid	IN	NUMBER	The identifier of the action.

Table 12 – 20 Set Control Item Approver Procedure Parameters (Page 1 of 2)

Parameter	Usage	Type	Description
funcmode	IN	VARCHAR2	The workflow function mode.
resultout	OUT	VARCHAR2	The process result.

Table 12 – 20 Set Control Item Approver Procedure Parameters (Page 2 of 2)

Set Notification Party Procedure

The procedure name is: **set_notification_party**

Use this procedure to specify persons to notify for approved and rejected issues and change documents.

Parameters

The following table lists the parameters that are used by the Set Notification Party procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type.
p_item_key	IN	VARCHAR2	The workflow item key.
p_status	IN	VARCHAR2	The control item status.
actid	IN	NUMBER	The identifier of the action.
funcmode	IN	VARCHAR2	The workflow function mode.
resultout	OUT	VARCHAR2	The process result.

Table 12 – 21 Set Notification Party Procedure Parameters (Page 1 of 1)

Project Status Report Workflow Extension

The project status report workflow extension enables you to customize the workflow processes for submitting, approving, and publishing a project status report.

You must determine how you want to submit, approve, and publish the report. See Overview of Project Status Reports, *Oracle Project Management User Guide.*

Processing

The default project status report workflow process calls the project status report workflow extension.

See Also

Designing Client Extensions: page 7 – 6

Location and Package Name

The extension is identified by the following items:

Item	Name
Body template	PAPRWFCB.pls
Specification template	PAPRWFCS.pls
Package	pa_report_workflow_client.

Table 12 – 22

Procedures

Start Workflow Procedure

The procedure name is **start_workflow**.

This procedure starts the workflow process for a project status report. You can modify this procedure to add company specific business rules which will get validated using this procedure.

The following table lists the parameters that Oracle Projects provides for the start_workflow procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	The workflow item type.
p_process_name	IN	VARCHAR2	Name of the workflow process.
p_item_key	IN	NUMBER	The workflow item key.
p_version_id	IN	NUMBER	Identifier of the version.
x_msg_count	OUT	NUMBER	The number of messages being sent.
x_msg_data	OUT	VARCHAR2	The content of the message.
x_return_status	OUT	VARCHAR2	The return status of the message.

Table 12 – 23 (Page 1 of 1) Start Workflow Parameters

Status Report Approver Procedure

The procedure name is: **set_report_approver**.

This procedure determines the approver for the project status report approval process. You can modify the procedure to add rules to determine who can approve a project. The default procedure returns the ID of the supervisor of the person who submitted the project status change.

The following table lists the parameters that Oracle Projects provides for the set_report_approver procedure.

Parameter	Usage	Type	Description
p_process	IN	VARCHAR2	Name of the workflow process.
p_item_key	IN	VARCHAR2	The workflow item key.

Parameter	Usage	Type	Description
actid	IN	NUMBER	Identifier of the action.
funcmode	IN	VARCHAR2	Workflow function mode.
resultout	OUT	VARCHAR2	Process result.

Table 12 – 24 (Page 2 of 2) Set Report Approver Parameters

Notification Party Procedure

The procedure name is set_report_notification_party.

This procedure determines which users receive workflow notifications when a project status report is submitted, approved, rejected, or published.

The following table lists the parameters that Oracle Projects provides for the set_report_notification_party procedure.

Parameter	Usage	Type	Description
p_item_type	IN	VARCHAR2	Name of the workflow process.
p_item_key	IN	VARCHAR2	The workflow item key.
p_status	IN	VARCHAR2	The report status.
actid	IN	NUMBER	Identifier of the action.
funcmode	IN	VARCHAR2	Workflow function mode.
resultout	OUT	VARCHAR2	Process result.

Table 12 – 25 (Page 1 of 1) Set Report Notification Party Parameters

PSI Client Extension

You can use a PSI client extension to derive an alternate column value, even if you have entered a column definition in the PSI Columns window. You can also use the extension to override the totals fields in the Project window.

To use a PSI client extension, you must:

- write the logic in a PL/SQL procedure and then store the procedure in the database
- define the column prompt for the column in the Project Status Inquiry Columns window

Running the PSI client extension will degrade the product's performance. Therefore, define your client extension procedures with as narrow a scope as possible.

The PSI Extensions Package

The extension is identified by the following items:

Item	Name
Body template	PAXVPS2B.pls
Specification template	PAXVPS2S.pls
Package	pa_client_extn_status

Table 12 – 26

The PSI Get Columns Procedure

The Get Columns procedure consists of three functions, one for each status folder (project, task, and resource):

- ProjCustomExtn
- TaskCustomExtn
- RsrcCustomExtn

The name of the Get Columns procedure is *getcols***.**

Each function has a parameter or "switch" that you can enable to run only that part of the client extension. You can run all, none, or any combination of the functions. By default, all three switches are disabled.

If you enable the Get Columns procedure, the Project Status window displays the column prompts defined in the PSI Columns window and the values calculated by the extension. Because the values calculated by the extension override values defined in the PSI Columns window, you do not need to enter a definition for a column whose value is calculated by a client extension.

> **Note:** If the procedure returns a NULL value, the Project Status window reads the value defined in the PSI Columns window.

Package.Procedure

The following table lists the parameters that Oracle Projects provides for the procedure *pa_client_extn_status.getcols*.

Parameter	Usage	Type	Description
X_project_id	IN	NUMBER	The identifier of the project
X_task_id	IN	NUMBER	The identifier of the task. This value is set to 0 if called for the project level columns
X_resource_list_member_id	IN	NUMBER	The identifier for the resource. This value is set to 0 if called for project or task level columns
X_cost_budget_type_code	IN	VARCHAR2	The identifier of the cost budget type displayed in PSI. This value is NULL when called from the resource status folder
X_rev_budget_type_code	IN	VARCHAR2	The identifier of the revenue budget type displayed in PSI. This value is NULL when called from the resource status folder
X_status_view	IN	VARCHAR2	The identifier of the status folder: PROJECTS, TASKS, or RESOURCES

Table 12 – 27 (Page 1 of 2) Get Columns Parameters

Parameter	Usage	Type	Description
X_pa_install	IN	VARCHAR2	The identifier of the Oracle Projects product installed: BILLING or COSTING. BILLING includes all default PSI columns. COSTING includes all but the actual revenue and revenue budget columns.
X_derived_col1 through X_derived_col3	OUT	VARCHAR2	Three alphanumeric derived columns. Each can have up to 255 characters. **Note:** Column 1 refers to the first column in both the PSI Columns and the Project Status windows, Column 2 refers to the second column in each window, etc.
X_derived_col_4 through X_derived_col_33	OUT	NUMBER	30 numeric derived columns. **Note:** Column 4 refers to the fourth column in both the PSI Columns and the Project Status windows, Column 5 refers to the fifth column in each window, etc.

Table 12 – 27 (Page 2 of 2) Get Columns Parameters

The PSI Get Totals Procedure

The PSI Get Totals procedure consists of two functions for PSI Project window totals functionality:

- Hide_Totals
- Proj_Tot_Custom_Extn

By default, these functions are disabled. If the Get Columns procedure is enabled for the Project window, then one of these functions automatically disables the Project window Totals button, unless the extension is modified.

If you enable the PSI Totals client extension, you can override the totals fields for all thirty numeric columns on the Project window for which you assign values to the OUT–parameters. The Project window displays NULL for any OUT–parameter that is not assigned a value.

For added flexibility, the Totals query actually selects and summarizes columns from a user–defined view, PA_STATUS_PROJ_TOTALS_V. By default, this view maps directly to the base view queried by the PSI Project window. Providing you maintain the same column names and data types for the first 34 columns, you may change the select statement, substitute literals for columns, and add unions to PA_STATUS_PROJ_TOTALS_V.

The name of the Get Totals procedure is *Get_Totals*.

Package.Procedure

The following table lists the parameters that Oracle Projects provides for the procedure *pa_client_extn_status.Get_Totals*.

Parameter	Usage	Type	Description
x_where_clause	IN	VARCHAR2	The where clause of the totals query statement
x_in_tot_column4 to x_in_tot_column33	IN	NUMBER	30 totals columns. The totals query assigns the totals that it returns to these columns. Column 4 refers to the fourth column in the PSI columns and the Project Status Inquiry windows.
x_in_tot_column4 to x_in_tot_column33	OUT	NUMBER	30 totals columns. The totals assigned by the Get_Totals procedure. Column 4 refers to the fourth column in the PSI columns and the Project Status Inquiry windows.

Table 12 – 28 (Page 1 of 2) Get Totals Parameters

Parameter	Usage	Type	Description
x_error_code	OUT	NUMBER	Error handling code. NOTE: A non–zero number invokes error processing by the PSI Project window and terminates totals processing.
x_error_message	OUT	VARCHAR2	User–defined error message. The non–zero error handling code and the user–defined message are displayed to the user in the event of an error. To facilitate debugging, the PSI Project window displays the totals returned by the totals query from PA_STATUS_PROJ_TOTALS_V.

Table 12 – 28 (Page 2 of 2) Get Totals Parameters

User–Defined Totals View

The following table lists the column names and data types that Oracle Projects provides for the user–defined totals view, PA_STATUS_PROJ_TOTALS_V.

> **Note:** While the first 34 column names and data types are required for the PSI Project window totals functionality, you may make modifications, such as changing the select statement or adding unions and new columns.

Column Name	Null	Type
PROJECT_ID	NOT NULL	NUMBER(15)
COLUMN1	NOT NULL	VARCHAR2(240)
COLUMN2	NOT NULL	VARCHAR2(240)
COLUMN3	NOT NULL	VARCHAR2(240)
COLUMN4 – COLUMN33	NOT NULL	NUMBER

Table 12 – 29 PA_STATUS_PROJ_TOTALS_V

The default select statement for PA_STATUS_PROJ_TOTALS_V is shown below:

```
CREATE or REPLACE FORCE VIEW PA_STATUS_PROJ_TOTALS_V
(PROJECT_ID,
```

```
COLUMN1,
COLUMN2,
COLUMN3,
COLUMN4...
)
AS SELECT
spg.project_id
spg.column1,
spg.column2,
spg.column3,
spg.column4...
FROM pa_status_proj_generic_v spg;
```

See Also

Project Summary Amounts, *Oracle Project Management User Guide*

PART IV: OPEN INTERFACES

CHAPTER

13

Oracle Projects Open Interfaces

This chapter describes the open interfaces in the Oracle Projects applications.

Transaction Import

Oracle Projects provides a single open interface, called Transaction Import. Transaction Import enables you to load transactions from external cost collection systems into Oracle Projects. Transaction Import creates pre–approved expenditure items from transaction data entered in external cost collection systems. Examples of external cost collection systems are:

- Timecard entry systems
- Expense report entry systems
- Supplier invoice entry systems, such as Oracle Payables
- Electronic data collection systems for asset usage (computer, printer, phone, etc.)
- Payroll systems that calculate complex transactions for benefits, overtime, and other labor charges
- Fixed assets systems that calculate depreciation charged to a project
- Manufacturing systems, such as Inventory and Work in Process

When loading transactions, Transaction Import creates expenditure batches, expenditures, and expenditure items. You can import costed or uncosted, accounted or unaccounted, and adjusted transactions into Oracle Projects.

You can use Transaction Import to import transactions that originate in any currency. The original currency and amount of each transaction is stored if the transaction currency is different from the project and / or functional currency.

This section describes how Transaction Import works. It also discusses how Transaction Import groups transactions to create expenditure batches. We also include information about the types of transactions you can load from external systems. Finally, we discuss how to view, process, and adjust the imported transactions in Oracle Projects.

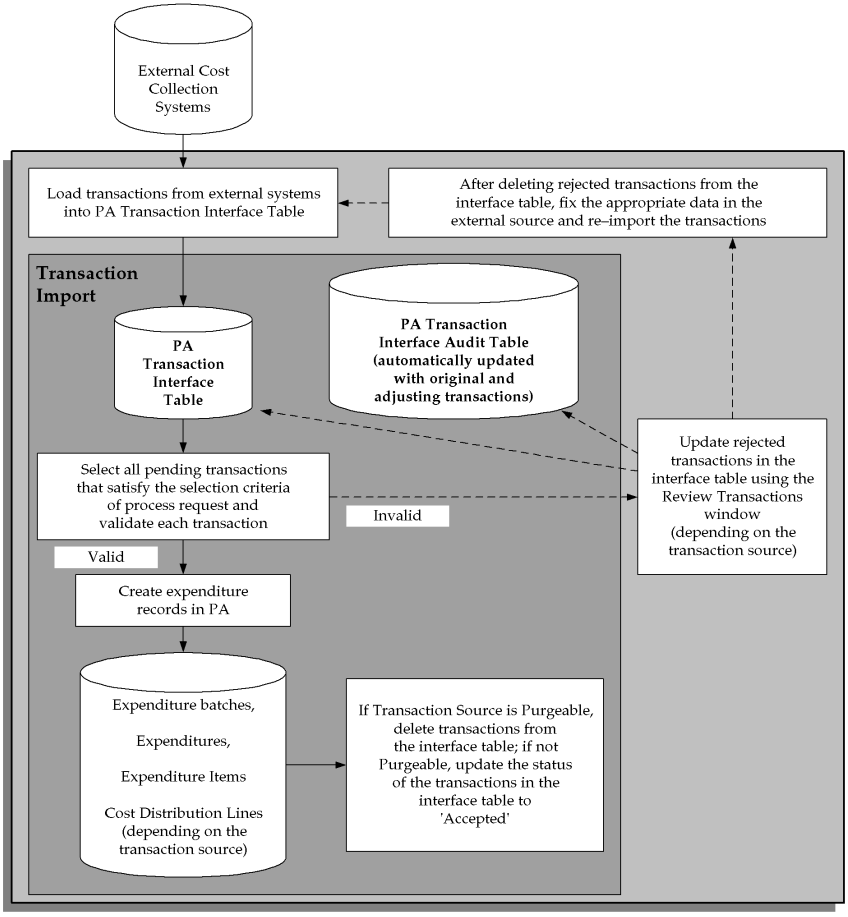
See Also

Transaction Import Interface: page 13 – 26

Expenditure Item Validation, *Oracle Project Costing User Guide*

Transaction Import Process Diagram

Figure 13 – 1 Transaction Import Process Diagram



Using Transaction Import

When you import transaction information from external cost collection systems, Oracle Projects records the transaction details and the source

of the imported transactions during transaction import. The PRC: Transaction Import process (also referred to as Transaction Import) validates the transaction information, reports any exceptions, and creates transactions for all of the valid transactions. Oracle Projects does not import a transaction more than once.

Populating the Interface Table

Transaction Import uses transaction data from your external system to create corresponding transactions in Oracle Projects.

Before you submit the PRC: Transaction Import process, you must populate the Transaction Interface table (PA_TRANSACTION_ INTERFACE_ALL) with records that you want to import.

To populate the table, you must write a custom feeder program to convert data into a standard data format that Transaction Import can read. Transaction Import can then convert your imported data into transactions in Oracle Projects.

Writing a Feeder Program

The type of environment from which you want to interface your data determines the type of feeder program you need to write. For example, you can use SQL*Loader, PL / SQL, or Pro*C to write a feeder program to interface transaction data from a non–Oracle system. Or, you can write a conversion program to interface historical data from your previous cost collection system.

Ensure that your transaction flat file has the appropriate information to populate PA_TRANSACTION_INTERFACE_ALL as indicated in the PA_TRANSACTION_INTERFACE_ALL Table Description. If a value is not required for a column, you may leave the column empty. See: PA_TRANSACTION_INTERFACE_ALL Table Description: page 13 – 27

Selecting an Import Utility

SQL*Loader is a powerful and easy–to–use tool that should be able to accommodate all of your import needs. However, depending on the complexity of your import program, you may also want to use Oracle's Pro* language products such as Pro*C, Pro*Cobol and Pro*Fortran to write the program.

Your import utility file must populate PA_TRANSACTION_ INTERFACE_ALL as indicated in the previous table description. Also,

you should code your file to populate the TRANSACTION_ SOURCE column in PA_TRANSACTION_INTERFACE_ALL with the Transaction Source code exactly as you defined it in the Transaction Sources window.

You must provide any information that the interface table requires that your external system does not provide. For example, if your external timecard system does not provide expenditure types, you must create at least one expenditure type and specify it in your control file.

Transaction Sources

When you submit Transaction Import, you must identify the source of the transactions that you want to import. The source can be any transaction source defined during implementation. You can also use transaction sources predefined by Oracle Projects.

The list of values for the transaction source parameter displays all of the transaction sources in the PA_TRANSACTION_SOURCES table. Any transaction source that has pending records in the Transaction Interface table are marked with an asterisk in the list of values.

Defining Transaction Sources

You define the source of transactions for Transaction Import in the Transaction Sources window. You can define an unlimited number of transaction sources. For each transaction source, you specify options that control how transactions are processed.

Use your import utility to enter this transaction source in the TRANSACTION_SOURCE column of the PA_TRANSACTION_ INTERFACE_ALL table. You then select the name in the Submit Request window when you want to import transactions from this source. See: Transaction Sources, *Oracle Projects Implementation Guide*.

Importing Transactions

After you populate the interface table, complete the following steps to import external transactions into Oracle Projects:

You use the Submit Request window to run Transaction Import.

▶ **To import transaction data into Oracle Projects:**

1. In the Navigator window choose Expenditures > Transaction Import > Import Transactions. Oracle Projects opens the Submit Request window and enters the PRC: Transaction Import request name.

   Alternately, you can navigate to the Submit Requests window and submit the PRC: Transaction Import process.

2. Choose the Transaction Source you want to process. (This field is required.)
3. Optionally identify a specific batch within the transaction source to process.
4. Choose Submit.

Correcting and Resubmitting Transactions

Use the Review Transactions window to review and resubmit rejected transactions or to create and submit new transactions. See: Resolving Import Exceptions: page 13 – 50

Output Reports

Transaction Import has two output reports:

- an exception report, which lists all rejected transactions
- a summary report of successfully imported transactions

See Also

Submitting Requests, *Oracle Projects Fundamentals*

Transaction Import Interface: page 13 – 26

Transaction Import Report, *Oracle Projects Fundamentals*

Types of Items That You Can Import

Using Transaction Import, you can import transactions with various expenditure type classes, as listed below.

- Labor
- Expense Reports
- Usages
- Inventory
- Work in Process
- Miscellaneous
- Supplier Invoices

You can import the transactions listed above from any transaction source associated with any expenditure type class.

Labor and Expense Report Transactions

When Project Resource Management is installed the import process associates labor and expense report transactions to scheduled work assignments as follows:

- If assignment information is provided by the external system, then that information is validated and imported as part of the transaction.
- If assignment information is not provided by the external system, then assignments are associated as follows:
  - If the resource for the transaction has only one available assignment, then the assignment is selected.
  - If the resource for the transaction has multiple available assignments, then the assignment with the earliest start date is selected.
  - If the resrouce for the transaction has no available assignments, then the transaction is imported as unscheduled.

  Note: You can override the association logic for resources with multiple available assignments using the transaction controls client extension. For information on transaction controls client extensions, see: page 8 – 21.

Assignments are considered available when the following conditions are met:

- The assignment resource equals the expenditure item resource.
- The assignment dates include the expenditure item date.
- The assignment status allows actual transactions.
- The schedule including the assignment is confirmed.

Unmatched Negative Transactions

You can import unmatched negative transactions. These transactions have a negative quantity and cost and do not reverse another transaction. Unmatched negative transactions are generally used for summary–level adjustments or to correct converted transactions.

Oracle Projects does not verify that an original transaction exists for unmatched negative transactions.

Exceptions

Overtime

Transaction Import does not import transactions with an expenditure type class of Overtime. However, you can load overtime from external systems by using an expenditure type class of Straight Time. To properly cost these transactions, you must either ensure that the transactions are loaded as costed, or use the Labor Costing extensions in Oracle Projects to properly calculate the overtime cost amounts.

See Also

Expenditure Type Classes, *Oracle Projects Implementation Guide*

Transaction Sources, *Oracle Projects Implementation Guide*

Loading Items as Costed or Uncosted

You can load uncosted items and costed items. The transaction source associated with the transaction specifies whether a transaction is costed or uncosted. If the Import Raw Cost Amounts option is selected for a transaction source, it indicates that the transactions have already been costed.

Uncosted Items Items for which only the quantity is provided. Oracle Projects costs these transactions like other transactions based on the cost multiplier and quantity.

Costed Items Items for which the quantity and transaction currency raw cost are provided. Oracle Projects does not recalculate the transaction currency raw cost of imported costed items.

With Oracle Projects, you can perform burdening and accounting on costed and uncosted items that you load via Transaction Import.

Loading Items as Accounted or Unaccounted

Each transaction source specifies whether items have already been accounted in the external system. Identifying items as accounted or unaccounted affects how Transaction Import processes the items. If the Raw Cost GL Accounted option is selected for a transaction source, it indicates that the transactions are accounted.

Unaccounted Items Items for which the appropriate GL account has not been determined. When loading unaccounted items, the Transaction Import process calls any transaction control extensions that you have defined. Cost calculation processes (distribute raw and burden costs) determine the cost amount (for uncosted items only) and the GL account to which the cost should be posted.

Accounted Items Items for which the functional currency raw cost amounts and GL accounts have already been determined and posted to GL by external systems. No processes within Oracle Projects will cost these transactions or transfer them to GL. When loading accounted items, Transaction Import creates cost distribution lines with a status of Received. Transaction Import also creates expenditure items and expenditures that are identified as accounted. If you import accounted items, you must provide the debit and credit code combination ID. When loading accounted transactions, Transaction Import will not call any extensions, create related items, or allow you to import related items.

Warning: If you import items with both the "GL Accounted" and "Allow Adjustments" options enabled, users will be able to adjust imported transactions that are already GL accounted. You may need to reconcile costs both between Oracle Projects and the external system, and between General Ledger and another general ledger application.

Loading Burden Transactions

You can import burden costs using the Transaction Import process. Depending on the definition of the transaction source, you can control how burden costs are imported and accounted. You can import the burdened costs as either a value on the expenditure item or as separate burden transaction expenditure items. Alternatively, you may choose not to import burden costs and allow Oracle Projects to calculate and store the burden costs as you have defined them in Oracle Projects.

Burden transactions have raw costs and quantities of zero and only burden amounts associated with the transactions. You identify burden transactions by assigning them an expenditure type class of Burden Transaction.

There is no predefined transaction source for burden transactions. You can create a new transaction source with a default expenditure type class of Burden Transaction and then use this transaction source to import burden transactions.

Controlling Import of Burden Transactions

Like the expenditure entry programs, Transaction Import allows burden transactions to be charged to projects that are not set up for burdening — that is, projects on which the associated project type costing information does not have the Burdened option enabled.

You can use Transaction Controls to prevent users from entering or importing burden transactions on a project.

Loading Project Manufacturing Costs

Oracle Projects predefines the transaction sources shown in the following table to enable you to import manufacturing resource costs from Oracle Manufacturing for the Project Manufacturing integration:

Transaction Source	Default Expenditure Type Class
Work in Process	Work in Process
Inventory	Inventory
Inventory Misc	Inventory

Table 13 – 1 Transaction sources and their default expenditure type classes

If you want to import manufacturing transactions from a non–Oracle manufacturing application, you must define your own transaction source.

Any transaction characterized by one of the transaction source and default expenditure type class combinations represented in the table above constitutes a manufacturing cost. However, you can use these transaction sources with other expenditure type classes. Note the following issues regarding Oracle Project Manufacturing transactions:

- Manufacturing transactions with a transaction source of Inventory or Work In Process are accounted for and interfaced to General Ledger by Oracle Manufacturing. Oracle Projects acts as the repository for these cost amounts but does not perform any accounting functions on them.
- Because they are transferred to Oracle Projects by sub–element (which maps to the expenditure type), multiple manufacturing transactions with a transaction source of either Inventory or Work In Process can use the same original system reference.
- You cannot adjust manufacturing costs in Oracle Projects, since all accounting for the costs is performed in Oracle Manufacturing. Any adjustments to these costs must originate in Oracle Manufacturing.

See Also

Integrating with Oracle Project Manufacturing, *Oracle Projects Fundamentals*

Loading Foreign Currency Transactions

Transaction Import enables you to import transactions that originate in any currency. This section describes how Transaction Import handles foreign currencies.

Currency Conversion Attributes for Imported Transactions

When transactions are imported that originated in a currency different from the functional currency or project currency, Oracle Projects must convert the transaction amount to those currencies.

To convert foreign currency transactions to the functional and project currencies, Oracle Projects must first determine the exchange rate type and exchange rate date.

To determine conversion attributes for foreign currency transactions imported by Transaction Import, Projects uses the logic shown below.

Each of the attributes is determined separately. That is, if a rate type is found in step one, but no rate date is found at that level, the rate type is used and the logic is followed to the next level to determine the rate date.

Case 1: Functional Currency Equals Project Currency

If the functional currency of the operating unit that incurred the cost (the expenditure operating unit) is equal to the functional currency of the operating unit that owns the project to which the cost is charged (the project operating unit), the following logic is used to determine the currency conversion attributes used in converting the transaction amounts from the transaction currency:

First, the functional currency attributes are determined as follows:

1. If user–entered conversion attribute is included in the transaction, that attribute is used for the conversion.
2. If user–entered attribute is not included in the transaction, the system looks for a default attribute for the task to which the transaction is charged.
3. If default conversion attribute does not exist for task, the system uses the default conversion attribute for the project to which the transaction is charged.
4. If there are no defaults entered at the project or task level, the default attribute is the attribute entered in the implementation options for the expenditure operating unit.

Note: For the Expense Report expenditure type, the conversion attributes entered in the implementation options are always used.

These attributes are used to obtain a conversion rate, which is used to convert the transaction currency amount to the functional currency. Since the functional currency is equal to the project currency, the project currency amount is equal to the functional currency amount.

This logic is illustrated in the following table:

Functional Currency Rate Type and Rate Date	Project Currency Rate Type and Rate Date
The following hierarchy is used: 1. User–entered value 2. Default value from the lowest task 3. Default value from the project 4. Default value from the expenditure operating unit's implementation options	The functional currency attributes are used.

Table 13 – 2 Functional currency equals project currency (Page 1 of 1)

Case 2: Functional Currency Does Not Equal Project Currency

If the functional currency for the transaction is not equal to the project currency, the following logic is used to determine the currency conversion attributes:

The functional currency attributes are determined as follows:

1. If a user–entered conversion attribute is included in the transaction, that attribute is used for the conversion.
2. If a user–entered attribute is not included in the transaction, the system uses the default attribute in the implementation options for the expenditure operating unit.

The attributes are used to obtain a conversion rate, which is used to convert the transaction currency amount to the functional currency.

The project currency attributes are determined as follows:

1. If user–entered conversion attribute is included in the transaction, that attribute is used for the conversion.

2. If user–entered attribute is not included in the transaction, the system looks for a default attribute for the task to which the transaction is charged.
3. If default conversion attribute does not exist for task, the system uses the default conversion attribute for the project to which the transaction is charged.
4. If there are no defaults entered at the project or task level, the default attribute is the attribute entered in the implementation options.
    - The default rate date is the implementation option for the expenditure operating unit.
    - The default rate type is the implementation option for the project operating unit.

The attributes are used to obtain a conversion rate, which is used to convert the transaction currency amount to the project currency.

This logic is illustrated in the following table:

Functional Currency Rate Type and Rate Date	Project Currency Rate Type and Rate Date
The following hierarchy is used: 1. User–entered value 2. Default value from the expenditure operating unit's implementation options	The following hierarchy is used: 1. User–entered value 2. Default value from the lowest task 3. Default value from the project 4. For the rate type, the default value from the project operating unit's implementation options. For the rate date, the default value from the expenditure operating unit's implementation options.

Table 13 – 3 Functional currency does not equal project currency (Page 1 of 1)

Functional Currency Raw Cost: Rounding Limit

When a transaction is imported as accounted, you must supply a value for ACCT_RAW_COST (functional raw cost). If the transaction currency is different from the functional currency, you must also supply the functional conversion attributes.

Transaction Import recalculates the functional raw cost, using the functional currency attributes you provide, to ensure that the imported functional raw cost and functional currency attributes are in agreement. The rounding limit (ACCT_EXCHANGE_ROUNDING_LIMIT) is used as a tolerance level when comparing the calculated and supplied figures.

If the difference between these two amounts is less than or equal to the tolerance limit, then Transaction Import accepts the transaction. Otherwise, the Transaction Import rejects the transaction.

Examples of this calculation are shown in the following table:

Column or Calculation	Example 1: Values Within Rounding Limit (transaction is accepted)	Example 2: Values Outside the Rounding Limit (transaction is rejected)
Transaction raw cost (DENOM_RAW_COST)	80 GBP	80 GBP
Functional raw cost (ACCT_RAW_COST)	100 USD	85 USD
Functional Exchange Rate (based on supplied currency attributes)	1.2375	1.2375
Rounding Limit (ACCT_EXCHANGE_ROUNDING_LIMIT)	10	10
Calculated functional raw cost (DENOM_RAW_COST * Functional Exchange Rate)	99 USD	99 USD
Difference between calculated and supplied functional raw cost	abs (100 – 99) = 1	abs (85 – 99) = 14

Table 13 – 4 (Page 1 of 1)

In Example 1, the calculated functional raw cost (99 USD) differs from the supplied functional raw cost (100 USD) by 1, which is less than the tolerance limit (10). Therefore, the transaction is accepted.

In Example 2, the values are the same as in Example 1, except that the supplied functional raw cost is 85 USD. This amount differs from the calculated functional raw cost (99 USD) by 14, which is more than the tolerance limit (10). Therefore, the transaction is rejected.

> **Note:** If the supplied ACCT_ROUND_LIMIT value is null, the rounding limit is zero.

Import Options (Transaction Source)

Transaction Import processes transactions based on the transaction source you select for each imported transaction. When you set up each transaction source, you select options that determine how transactions are processed by Transaction Import.

Following are some of the fields and actions you can control when you choose transaction source options:

- the default expenditure type class
- whether Projects calculates raw cost amounts
- whether Projects calculates burden amounts
- whether Projects interfaces amounts to GL and AP
- whether Projects imports the expenditure organization for employee transactions
- whether to allow manual adjustments before transaction import
- whether duplicate reference IDs are allowed within a transaction source
- whether the transaction can be reversed or adjusted after it is imported
- whether Projects calculates MRC reporting currency amounts

For a detailed description of all transaction source options, see: Transaction Sources, *Oracle Projects Implementation Guide*.

Grouping Transactions into Expenditure Batches and Expenditures

This section describes how Transaction Import groups transactions into expenditure batches and expenditures.

When you load transactions into the interface table from an external system, Oracle Projects requires that you specify the following information for each transaction:

- Transaction source
- Batch name
- Expenditure ending date
- Employee name or Organization
- Expenditure type class (if this information is not provided for the transaction, the value defaults to the expenditure type class assigned to the transaction source during implementation)
- The following currency attributes, if foreign currencies are used:
  - transaction currency
  - functional currency conversion rate date
  - functional currency conversion rate type
  - functional currency conversion rate

Transaction Import groups all of the transactions processed during an interface run into expenditures and expenditure batches in the following manner.

Attention: If the employee number is specified, Transaction Import ignores any value for the organization and derives the organization value based on the employee's assignment.

An exception to this is if the Import Employee Organization option is selected for the transaction source.

Straight Time and Expense Reports

If the transaction source of the transactions being processed is defined with an expenditure type class of *Straight Time* or *Expense Reports*, the transactions are grouped into expenditures and expenditure batches based on the following information:

- Transaction source
- Expenditure type class
- Batch name

- Employee number
- Expenditure ending date
- Expenditure Organization: This information is used if the ALLOW_EMP_ORG_OVERRIDE flag in the transaction sources table is set to ”Y”
- Additional grouping criteria, provided by the user, using the following columns:
  - ORIG_EXP_TXN_REFERENCE1
  - USER_ORIG_EXP_TXN_REFERENCE
  - VENDOR_NUMBER
  - ORIG_EXP_TXN_REFERENCE2
  - ORIG_EXP_TXN_REFERENCE3
- The following currency attributes, if applicable:
  - transaction currency
  - functional currency conversion rate date
  - functional currency conversion rate type
  - functional currency conversion rate

Each unique batch name becomes an expenditure batch, and each unique expenditure type class, employee number, and expenditure ending date combination becomes an expenditure within the expenditure batch. The ending date of the expenditure batch is set to the maximum ending date of all the expenditures created within that batch.

An employee number is required for all transactions with an expenditure type class Straight Time or Expense Reports. Transactions with any other expenditure type classes do not require an employee number.

All Other Expenditure Type Classes

If the transaction source of the transactions being processed is defined with an expenditure type class of *Usages*, *Miscellaneous Transactions*, *Burden Transactions*, *Inventory*, or *Work in Process*, the key information in the interface table used in grouping transactions into expenditures and expenditure batches is as follows:

- Transaction source

- Expenditure type class
- Batch name
- Employee number (optional)
- Expenditure organization name
- Expenditure ending date
- Additional grouping criteria, provide by the user, using the following columns:
  - ORIG_EXP_TXN_REFERENCE1
  - USER_ORIG_EXP_TXN_REFERENCE
  - VENDOR_NUMBER
  - ORIG_EXP_TXN_REFERENCE2
  - ORIG_EXP_TXN_REFERENCE3
- The following currency attributes, if applicable:
  - transaction currency
  - functional currency conversion rate date
  - functional currency conversion rate type
  - functional currency conversion rate

Each unique batch name becomes an expenditure batch, and each unique expenditure type class, employee number, organization, and expenditure ending date combination becomes an expenditure within the expenditure batch. The ending date of the expenditure batch is set to the maximum ending date of all the expenditures created within that batch.

Transaction Import Example: Labor and Expense by Employee Number

In this example, all imported expenditures are in the functional currency. Therefore, currency attributes are ignored in grouping expenditure items.

You load the following transactions (expenditure items) into the interface table. The transaction source of Site1 has expenditure type classes of Straight Time and Expense Reports.

Trx Number	Trx Source	Expenditure Type Class	Batch Name	Employee Number	Expenditure Ending Date
1	Site1	Straight Time	L1	1000	02–OCT–95
2	Site1	Straight Time	L1	1000	25–SEP–95
3	Site1	Expense Reports	L1	1000	25–SEP–95
4	Site1	Expense Reports	L1	1001	09–OCT–95
5	Site1	Straight Time	L2	1001	09–OCT–95
6	Site1	Straight Time	L2	1001	09–OCT–95

Table 13 – 5 (Page 1 of 1)

If you submit Transaction Import for the transaction source of Site1 and do not specify a specific batch to process (pick all transactions with a transaction source of Site1), then Transaction Import will process all six of the above transactions.

Assuming that all of the transactions in this example are valid, then Oracle Projects creates two expenditure batches—L1 and L2:

Batch Name	Transaction Number	Expenditure Type Class	Employee Number	Expenditure Ending Date
L1	1	Straight Time	1000	02–OCT–95
	2	Straight Time	1000	25–SEP–95
	3	Expense Reports	1000	25–SEP–95
	4	Expense Reports	1001	09–OCT–95
L2	5	Straight Time	1001	09–OCT–95
	6	Straight Time	1001	09–OCT–95

Table 13 – 6 (Page 1 of 1)

Since the transaction source has expenditure type classes of Straight Time and Expense Reports, Transaction Import groups the transactions by employee, expenditure ending date, and expenditure type class when creating expenditures. Therefore, the resulting expenditures are as follows:

Batch Name	Transaction Number	Expenditure Type Class	Employee Number	Expenditure Ending Date
L1	1	Straight Time	1000	02–OCT–95
	2	Straight Time	1000	25–SEP–95
	3	Expense Reports	1000	25–SEP–95
	4	Expense Reports	1001	09–OCT–95
L2	5,6	Straight Time	1001	09–OCT–95

Table 13 – 7 (Page 1 of 1)

Notice that even though transactions 2 and 3 were for the same employee and the same ending date, Oracle Projects created two expenditures. Transactions with different expenditure type classes are imported into different expenditure batches. Different batch names will also result in the creation of different expenditure batches, even if they contain transactions for the same employee and ending date.

Since the ending date of the expenditure batch created is equal to the maximum ending date of the expenditures created within that batch, the batch ending dates for our example are as follows:

Batch Name	Expenditure Ending Date
L1	09–OCT–95
L2	09–OCT–95

Table 13 – 8 (Page 1 of 1)

Transaction Import Example: Usage

In this example, all imported expenditures are in the functional currency. Therefore, currency attributes are ignored in grouping expenditure items.

You load the following transactions into the interface table; the transaction source of Usage has an expenditure type class of Usages. The grouping logic is slightly different for usage items, because usage expenditures can be created for an employee or an organization.

> **Note:** You do not need to enter an employee number for usage transactions.

Trx Number	Txn Source	Expenditure Type Class	Batch Name	Employee Number	Organi–zation	Expenditure Ending Date
1	Usage	Usages	U1	1000	West	02–OCT–95
2	Usage	Usages	U1	1000	East	02–OCT–95
3	Usage	Usages	U1		West	02–OCT–95
4	Usage	Usages	U1		Midwest	02–OCT–95

Table 13 – 9 (Page 1 of 1)

Since all of these transactions have the same batch name, Oracle Projects creates only one expenditure batch. For usage items, Transaction Import groups transactions by employee, organization, and expenditure ending date when creating expenditures. Therefore, the resulting expenditures after import would be as follows:

Batch Name	Trx Number	Expenditure Type Class	Employee Number	Organization	Expenditure Ending Date
U1	1, 2	Usages	1000	*Employee Org*	02–OCT–95
	3	Usages		West	02–OCT–95
	4	Usages		Midwest	02–OCT–95

Table 13 – 10 (Page 1 of 1)

Notice that transactions (1) and (2) appear in the same expenditure because they were for the same employee / expenditure ending date, even though the organization name specified for both is different. If a transaction specifies an employee number, Transaction Import ignores any value for Organization and derives the organization value based on the employee's assignment (if the Import Employee Organization option is not used).

Also note that even if employee 1000's organization assignment were West, the resulting expenditures would still be the same. Transaction Import never groups usage transactions for an employee into the same expenditure as usage transactions for an organization.

Viewing Transactions in Oracle Projects

Transaction Import loads transactions as pre–approved expenditure items. Expenditure batches are created with a status of Released. A status of Released indicates that the expenditure batch is fully approved and ready for cost distribution.

> **Note:** All transactions that have already been accounted for in external systems, including manufacturing transactions, are loaded as costed transactions. These transactions are created with cost distribution lines and a status of Received.

You can view imported expenditure batches and associated expenditures and expenditure items using the Expenditure Inquiry and Expenditure Batches windows in Oracle Projects.

Expenditure Batch Names

The expenditure batch name within Oracle Projects is created as a concatenation of the batch name and expenditure type class entered in the transaction interface table and the interface ID. For example, an expenditure batch name may appear as follows: B1ST101.

B1 is the batch name loaded from the external system. ST is the expenditure type class ('ST' for Straight Time). 101 is the interface ID generated when you run Transaction Import.

The maximum length of the expenditure batch name is 20 characters (10 for the batch name, 3 for the expenditure type class, and 7 for the interface ID). The interface ID is an Oracle sequence that resets to 1 after 9999999. If a duplicate expenditure batch name results from resetting the interface ID to 1, change the batch name of the entire batch.

Viewing Transactions in the Audit Report

To see detailed information on successfully imported expenditure items, use the following information as parameters for the AUD: Pre–Approved Expenditure Entry Audit Report. The information for these parameters is displayed in the Transaction Import output report.

- Expenditure batch
- Employee name that corresponds to the user ID of the person who submitted Transaction Import

Identifying the ”Entered By” User for Reporting Purposes

For viewing imported transaction online, or for using the Entered By parameters in reports such as the Pre–Approved Expenditures Entry Audit Report, use the employee name that corresponds to the user ID of the person who submitted the process as the entered by person.

Suggestion: You may want to create a new user to run Transaction Import with a unique name, such as TRX IMPORT USER, so you can easily identify and report imported transactions.

Adjusting Imported Transactions in Oracle Projects

You can adjust imported transactions in Oracle Projects, if the transaction source allows this type of change. See: Expenditure Adjustments, *Oracle Project Costing User Guide* and Transaction Sources, *Oracle Projects Implementation Guide*.

> **Note:** Raw cost values for transactions that were already costed when loaded into Oracle Projects will not be changed if you mark the item for cost recalculation.

Uniquely Identifying Transactions

You can uniquely identify imported transactions by the transaction source and the original transaction reference, if you do not allow duplicate system references for the transaction source. You can review this information in the Expenditure Items window, which you can access from the Expenditure Inquiry window.

Processing Imported Transactions

Oracle Projects processes imported transactions just as it processes transactions entered using the expenditure entry forms. The imported transactions that are not accounted (as specified for the transaction source) are processed in the appropriate cost distribution program. If expenditure items are billable and charged to a contract project, they are also processed during revenue and invoice generation. Accounting transactions are then interfaced to other Oracle Applications.

Purging Imported Transactions

You can purge imported transactions from the interface table either automatically or manually:

- To purge imported transactions automatically, you specify that a particular transaction source is purgeable.
- To purge imported transactions manually, use SQL*Plus to remove the records from the interface table.

Transaction Import Interface

This section includes a detailed description of the Transaction Import interface table, PA_TRANSACTION_INTERFACE_ALL. It also describes the validation Oracle Projects performs for imported transactions. This section also describes how to resolve import exceptions.

Transaction Import Validation

You use an import utility to load transaction information into the interface table (PA_TRANSACTION_INTERFACE_ALL) for each transaction you want to create. You can load the table directly from your external system, or you can fill in some values using SQL*Plus.

Transaction Import validates your data for compatibility with Oracle Projects by ensuring that the columns in the interface table reference the appropriate and active values and columns in Oracle Projects.

Validating Expenditure Items

Transaction Import validates all items within an expenditure before it creates an expenditure. If at least one item in an expenditure fails the validation, Oracle Projects rejects all items in the expenditure. The item that failed is marked with a rejection reason; all other items in the expenditure are marked as rejected without a reason.

Transaction Import detects only one error per transaction each time you run the import process. If a single transaction has multiple errors, you will need to run Transaction Import more than once to discover all the errors.

You can correct rejected transactions using the Review Transactions window. After you make your corrections, you can validate the revised information by resubmitting the corrected transactions from the same window. See: Resolving Import Exceptions: page 13 – 50.

If Transaction Import detects errors during the validation process, you do not need to correct all rejected items to save your transaction information. You need to correct all items, however, before you can successfully import your transactions.

Validating and Loading Transactions

Transaction Import validates data before importing it, to ensure that your transactions contain the appropriate data for Oracle Projects. For a list of the validation criteria, see: Expenditure Item Validation, *Oracle Project Costing User Guide*.

> **Note:** Detailed information on additional column validation is contained in the section: The Transaction Import Interface Table: page 13 – 27.

Target Expenditure Tables

The Transaction Import program validates all required transaction data in this table. If the transaction data is valid, Transaction Import creates transactions (expenditure items) from the information in the interface table and places the transaction information in the following expenditure tables:

- PA_EXPENDITURE_GROUPS_ALL
- PA_EXPENDITURES_ALL
- PA_EXPENDITURE_ITEMS_ALL
- PA_COST_DISTRIBUTION_LINES_ALL
- PA_EXPENDITURE_COMMENTS

The Transaction Import Interface Table

The Transaction Import interface table (PA_TRANSACTION_INTERFACE_ALL) is the table you populate to import transactions from external sources into Oracle Projects. For a full description of the Transaction Import interface table, including foreign keys and database triggers, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

NULL and NOT NULL Columns

The table description for PA_TRANSACTION_INTERFACE_ALL in Oracle eTRM indicates whether each column in the Transaction Import interface table is a NULL or NOT NULL column.

NOT NULL columns

You must enter values for all NOT NULL columns to successfully import an expenditure item.

NULL Columns

A NULL column is a column in the interface table that does not require a value. There are two types of NULL columns:

- Some NULL columns are required only for some types of transactions. For example, for usage items, the NON_LABOR_RESOURCE column must be populated. We mark these columns as Conditionally Required. See: Conditionally Required: page 13 – 28.
- Some NULL columns should never be populated because they are used by the Transaction Import program. These columns are called System Assigned Columns. See System Assigned: page 13 – 29.

Other Column Requirements

Details about the requirements that you need to consider when you populate the interface table are elaborated below:

Conditionally Required

Conditionally required columns may require a value, depending on the value in another column.

For example, if you are importing a usage expenditure item, the value of SYSTEM_LINKAGE is Usages. When the value of SYSTEM_LINKAGE is Usages, you must supply a value for the NON_LABOR_RESOURCE and NON_LABOR_RESOURCE_ORGANIZATION columns. (These columns are not required for labor expenditure items.) Therefore, the Comments column for NON_LABOR_RESOURCE contains the words ”Required for usage items.”

As another example, several columns are required only if the transaction has been accounted and interfaced to GL. This criterion is determined as follows:

1. The column TRANSACTION_SOURCE must contain a valid transaction source.

2. Each transaction source in Oracle Projects has an flag that indicates whether transactions have been GL accounted before being imported.

Optional

Columns marked Optional are for optional transaction information tracking.

You can use these columns to import additional information for the transactions that Transaction Import creates. Transaction Import imports the data that you load into these optional columns, provided that the information passes the validation checks.

System Assigned

Oracle Projects assigns values to the system–assigned columns during the import process.

Attention: Your import file must leave these columns blank.

Transaction Interface Control Table

Oracle Projects uses the PA_TRANSACTION_XFACE_CTRL_ALL table to control processing of transactions by the Transaction Import program. You must not insert or update records in this table directly. This table is populated by database triggers when you load or update the PA_TRANSACTION_INTERFACE table.

PA_TRANSACTION_INTERFACE_ALL Column Requirements

The following table shows information about column requirements for some of the columns in the PA_TRANSACTION_INTERFACE_ALL table. For column descriptions, datatype, and null / not null information, refer to Oracle eTRM, which is available on Oracle*MetaLink*.

Column Name	Requirements
TRANSACTION_SOURCE: page 13 – 32	
BATCH_NAME: page 13 – 33	

Table 13 – 11 (Page 1 of 4) PA_TRANSACTION_INTERFACE_ALL Requirements

Column Name	Requirements
EXPENDITURE_ENDING_DATE: page 13 – 33	
EMPLOYEE_NUMBER: page 13 – 33	
ORGANIZATION_NAME: page 13 – 34	
EXPENDITURE_ITEM_DATE: page 13 – 34	
PROJECT_NUMBER: page 13 – 34	
TASK_NUMBER: page 13 – 35	
EXPENDITURE_TYPE: page 13 – 35	
NON_LABOR_RESOURCE: page 13 – 35	Required for usage items
NON_LABOR_RESOURCE_ORG_NAME: page 13 – 36	Required for usage items
QUANTITY: page 13 – 36	
RAW_COST: page 13 – 36	Required for costed items. Otherwise optional
EXPENDITURE_COMMENT: page 13 – 36	Optional
TRANSACTION_STATUS_CODE: page 13 – 37	
TRANSACTION_REJECTION_CODE: page 13 – 37	System assigned
EXPENDITURE_ID: page 13 – 38	System assigned
ORIG_TRANSACTION_REFERENCE: page 13 – 38	
ATTRIBUTE_CATEGORY: page 13 – 38	Optional
ATTRIBUTE1 through ATTRIBUTE10: page 13 – 38	Optional
RAW_COST_RATE: page 13 – 39	Optional
INTERFACE_ID: page 13 – 39	System assigned
UNMATCHED_NEGATIVE_TXN_FLAG: page 13 – 39	Optional
EXPENDITURE_ITEM_ID: page 13 – 40	System assigned
ORG_ID: page 13 – 40	
DR_CODE_COMBINATION_ID: page 13 – 40	Required for transactions already accounted for in GL. (Each transaction source indicates whether transactions are GL accounted or not.)
CR_CODE_COMBINATION_ID: page 13 – 40	Required for transactions already accounted for in GL
CDL_SYSTEM_REFERENCE1: page 13 – 41	Required for transactions already accounted for in GL
CDL_SYSTEM_REFERENCE2: page 13 – 41	Required for transactions already accounted for in GL
CDL_SYSTEM_REFERENCE3: page 13 – 41	Required for transactions already accounted for in GL
GL_DATE: page 13 – 41	Required for transactions already accounted for in GL
BURDENED_COST: page 13 – 42	Required for burden transactions
BURDENED_COST_RATE: page 13 – 42	Required for burden transactions
SYSTEM_LINKAGE: page 13 – 42	Optional

Table 13 – 11 (Page 2 of 4) PA_TRANSACTION_INTERFACE_ALL Requirements

Column Name	Requirements
TXN_INTERFACE_ID: page 13 – 42	System assigned
USER_TRANSACTION_SOURCE: page 13 – 43	Required if TRANSACTION_SOURCE is not populated
RECEIPT_CURRENCY_AMOUNT: page 13 – 43	Required if: – SYSTEM_LINKAGE is Expense Reports – and the item is uncosted – and RECEIPT_CURRENCY_CODE is not null and is different from TRANSACTION_CURRENCY_CODE
RECEIPT_CURRENCY_CODE: page 13 – 43	Optional
RECEIPT_EXCHANGE_RATE: page 13 – 43	Required if RECEIPT_CURRENCY_AMOUNT and RECEIPT_CURRENCY_CODE are both not null.
DENOM_CURRENCY_CODE: page 13 – 44	Required
DENOM_RAW_COST: page 13 – 44	Required for costed items
DENOM_BURDENED_COST: page 13 – 44	Required for burden transactions
ACCT_RATE_DATE: page 13 – 44	Required for accounted transactions if functional currency and DENOM_CURRENCY_CODE are not the same
ACCT_RATE_TYPE: page 13 – 45	Required for accounted transactions if functional currency and DENOM_CURRENCY_CODE are not the same
ACCT_EXCHANGE_RATE: page 13 – 45	Required: (1) for accounted transactions if functional currency and DENOM_CURRENCY_CODE are not the same, or (2) if ACCT_RATE_TYPE is User
ACCT_RAW_COST: page 13 – 45	Required for accounted transactions
ACCT_BURDENED_COST: page 13 – 46	Optional
ACCT_EXCHANGE_ROUNDING_LIMIT: page 13 – 46	Optional
PROJECT_CURRENCY_CODE: page 13 – 46	System assigned
PROJECT_RATE_DATE: page 13 – 46	Optional
PROJECT_RATE_TYPE: page 13 – 47	Optional
PROJECT_EXCHANGE_RATE: page 13 – 47	Required if PROJECT_RATE_TYPE is User
ORIG_EXP_TXN_REFERENCE1: page 13 – 47	Populated with DIST.INVOICE_ID by Interface Supplier Invoices process.
ORIG_EXP_TXN_REFERENCE2: page 13 – 47	Used for additional grouping.
ORIG_EXP_TXN_REFERENCE3: page 13 – 48	Used for additional grouping.
ORIG_USER_EXP_TXN_REFERENCE: page 13 – 48	Populated with INV.INVOICE_NUM by Interface Supplier Invoices process.
VENDOR_NUMBER: page 13 – 48	Populated with INV.VENDOR_ID by Interface Supplier Invoices process.

Table 13 – 11 (Page 3 of 4) PA_TRANSACTION_INTERFACE_ALL Requirements

Column Name	Requirements
OVERRIDE_TO_ORGANIZATION_NAME: page 13 – 48	Ignored unless the Import Employee Organization option is set to Y in the Transaction Source
REVERSED_ORIG_TXN_REFERENCE: page 13 – 48	Optional
BILLABLE_FLAG: page 13 – 49	Optional
PERSON_BUSINESS_GROUP_NAME: page 13 – 49	Optional
CREATED_BY: page 13 – 49	Standard Who Column
CREATION_DATE: page 13 – 49	Standard Who Column
LAST_UPDATED_BY: page 13 – 50	Standard Who Column
LAST_UPDATE_DATE: page 13 – 50	Standard Who Column

Table 13 – 11 (Page 4 of 4) PA_TRANSACTION_INTERFACE_ALL Requirements

The following section shows further detail about each PA_TRANSACTION_INTERFACE_ALL column, including validation and destination information.

TRANSACTION_SOURCE

Enter an implementation–defined transaction source code that classifies the transaction. This transaction source, along with the original transaction reference, identifies the source of transactions loaded into Oracle Projects from an external system.

If a transaction source is defined with the Import Raw Cost Amounts option enabled, then a raw cost amount must exist for the transaction.

If the transaction source is defined with the Purge After Import option enabled, then the transaction will be purged from the table when the import process has completed.

See: Transaction Sources, *Oracle Projects Implementation Guide*.

> **Note:** This column is for internal use only; you cannot view values stored in this column from any Oracle Projects windows. You must enter this internal code or a value in USER_TRANSACTION_SOURCE to specify the transaction source.

Validation: The transaction source you enter must be a valid transaction source. You can obtain a list of valid transaction sources from PA_TRANSACTION_SOURCES.TRANSACTION_SOURCE.

Destination: PA_EXPENDITURE_GROUPS_ALL. TRANSACTION_SOURCE and

PA_EXPENDITURE_ITEMS.TRANSACTION_SOURCE. The transaction source information is denormalized for performance optimization.

BATCH_NAME

An expenditure batch is a group of expenditures loaded into the interface table. All transactions in a batch must have the same transaction source.

The batch name is used to derive part of the expenditure batch name used in the expenditure tables. The expenditure batch name is created by Oracle Projects by combining the following three items from the transaction interface table:

- batch name (user–supplied)
- expenditure type class (user–supplied)
- interface ID (system–generated)

Note: You must enter a batch name of twenty characters or less.

Validation: None

Destination: PA_EXPENDITURE_GROUPS_ALL.EXPENDITURE_GROUP

EXPENDITURE_ENDING_DATE

Enter the date of the last day of the expenditure week for this transaction. All transactions in an expenditure must be on or before the expenditure ending date. In addition, all timecard items must be within the expenditure week date range. The maximum expenditure ending date of all expenditure items processed in a batch becomes the expenditure batch ending date.

Validation: Valid week ending date based on the expenditure cycle start day defined in Implementation Options.

Destination: PA_EXPENDITURES_ALL.EXPENDITURE_ENDING_DATE

EMPLOYEE_NUMBER

Enter the number of the employee who incurred the charge for this transaction. This column must be populated for labor and expense report items, but is optional for other expenditure type classes.

Validation: Must be a valid employee number in PER_PEOPLE_F.EMPLOYEE_NUMBER

If a business group is specified in the PERSON_BUSINESS_GROUP_NAME field, then the employee must be defined in that business group.

Destination: PA_EXPENDITURES_ALL.INCURRED_BY_PERSON_ID

ORGANIZATION_NAME

Enter the name of the organization that incurred the charge for this transaction. If employee number is provided, then this column can be null, in which case Transaction Import derives this value from the employee organization. If you provide both an employee and an organization, Oracle Projects uses the employee information to derive the organization (if the Import Employee Organization option is not used). Transaction Import uses the last employee assignment in the expenditure period to derive the employee organization.

Validation: Must be a valid organization in PER_ORGANIZATION_UNITS.NAME

Destination: PA_EXPENDITURES_ALL.INCURRED_BY_ORGANIZATION_ID

EXPENDITURE_ITEM_DATE

Enter the date on which this transaction occurred.

Validation: The expenditure item date must be on or before the expenditure ending date. Also, the expenditure item date of timecard items must fall within the expenditure week as defined by the expenditure ending date.

Destination: PA_EXPENDITURE_ITEMS_ALL.EXPENDITURE_ITEM_DATE

PROJECT_NUMBER

Enter the number of the project this transaction is charged to.

Validation: Must be a valid project number in PA_PROJECTS.SEGMENT1 and PA_PROJECTS_EXPEND_V; project must have a

project status that allows new transactions; project must not be a project template; and project must allow charges from your operating unit (if multiple organization support is enabled).

Destination: None

TASK_NUMBER

Enter the number of the task this transaction is charged to.

Validation: Must be a valid task number in PA_TASKS.TASK_NUMBER for the project number specified; and task must be a lowest task that allows charges.

Destination: PA_EXPENDITURE_ITEMS_ALL.TASK_ID

EXPENDITURE_TYPE

Enter the expenditure type that classifies the type of charge for this transaction.

Validation: This expenditure type must be a valid expenditure type in PA_EXPENDITURE_ TYPES.EXPENDITURE_TYPE. The expenditure type and expenditure type class combination must exist as an active combination in the PA_EXPEND_TYP_SYS_LINKS table. You cannot import expenditure items with a expenditure type class of Supplier Invoices.

Destination: PA_EXPENDITURE_ITEMS_ALL. EXPENDITURE_TYPE

NON_LABOR_RESOURCE

Enter the non–labor resource utilized for this transaction. This column is populated only for usage items.

Validation: This non–labor resource must be a valid non–labor resource in PA_NON_LABOR_ RESOURCES.NON_LABOR_RESOURCE and must be a resource classified by the specified expenditure type.

Destination: PA_EXPENDITURE_ITEMS_ALL.NON_LABOR_ RESOURCE

NON_LABOR_RESOURCE_ORG_NAME

Enter the name of the organization owning the non–labor resource utilized for the transaction. This column is populated only for usage items.

Validation: Must be a valid non–labor resource owning organization in PA_NON_LABOR_RESOURCE_ORGS.ORGANIZATION_ID for the specified non–labor resource.

Destination: rPA_EXPENDITURE_ITEMS_ALL.ORGANIZATION_ID

QUANTITY

Enter the number of units for the transaction based on the unit of measure defined for the expenditure type. If the transaction is a multi–currency transaction and the expenditure type Unit of Measure is currency, then the quantity is the project currency quantity.

For burden transactions, this value must equal zero.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.QUANTITY

RAW_COST

Enter the total raw cost of the transaction. If the transaction is a multi–currency transaction, then this is the project currency raw cost amount.

For burden transactions, this value must equal zero.

Validation: If the transaction source is defined with the Import Raw Cost Amounts option enabled, a raw cost amount must exist. If the Import Raw Cost Amounts option is not enabled for the transaction source, then this column value is ignored.

Destination: PA_EXPENDITURE_ITEMS_ALL.RAW_COST

EXPENDITURE_COMMENT

Enter the description that you want to assign to the expenditure item created from this transaction.

Validation: None

Destination: PA_EXPENDITURE_COMMENTS. EXPENDITURE_COMMENT

TRANSACTION_STATUS_CODE

You must set this value to *P* for transactions you want to import.

If TRANSACTION_STATUS_CODE is set to *A* after Transaction Import completes, this indicates that the TRANSACTION_SOURCE entered is not defined with the Purge After Import option enabled, and you must delete the item manually. See: Transaction Sources, *Oracle Projects Implementation Guide*.

If an item is rejected, the rejection reason code will be generated in the TRANSACTION_REJECTION_CODE column.

The status codes are listed in the following table:

Status Code	Description
PO	Rejected in post–import
P	Pending
R	Rejected
PR	Rejected in pre–import
I	Imported
A	Accepted

Table 13 – 12 (Page 1 of 1)

Validation: Lookup codes for this column are stored in the PA_LOOKUPS table under the lookup type of TRANSACTION STATUS.

Destination: None

TRANSACTION_REJECTION_CODE

This column is populated by a system–defined code indicating why the transaction was rejected by the Transaction Import program. For a list of codes, see: Resolving Import Exceptions: page 13 – 50.

Validation: This column is system assigned. Lookup codes for this column are stored in the PA_LOOKUPS table under the lookup type of TRANSACTION REJECTION REASON.

Destination: None

EXPENDITURE_ID

This column is populated by a system–defined value to identify the transactions grouped into an expenditure.

Validation: This column is system assigned.

Destination: PA_EXPENDITURES_ALL.EXPENDITURE_ID

ORIG_TRANSACTION_REFERENCE

Enter a reference to the original item imported from an external system via the Transaction Import program. Unless the transaction source allows duplicate references, this reference, along with the transaction source, uniquely identifies the original transaction.

Validation: An expenditure item must not already exist with the same identifier values as the transaction if the transaction source does not allow duplicate system reference values.

Destination: PA_EXPENDITURE_ITEMS_ALL. ORIG_TRANSACTION_REFERENCE

ATTRIBUTE_CATEGORY

Enter the descriptive flexfield category for the descriptive flexfield information you want to import.

Validation: Validated using the standard AOL application programming interface (API) for validating attribute categories.

Destination: PA_EXPENDITURE_ITEMS_ALL. ATTRIBUTE_CATEGORY

ATTRIBUTE1 through ATTRIBUTE10

Enter the descriptive flexfield information that you want to import for a transaction (expenditure item). The structure of the information you enter in these columns (datatypes, value sets) should match the structure of the descriptive flexfield segments you have defined for your transaction or you will experience validation problems when you try to access the information in the expenditure entry forms.

Validation: Validated as expenditure item descriptive flexfield attributes, using the standard AOL application programming interface (API) for validating descriptive flexfields. **NOTE:** You must populate this field with the *attribute ID* (code) rather than the *meaning*. The meaning will not pass the validation.

NOTE: The Transaction Import process will validate descriptive flexfield attributes only if the Attribute Category field is populated.

Destination: PA_EXPENDITURE_ITEMS_ALL. ATTRIBUTE1 through ATTRIBUTE10

RAW_COST_RATE

Enter the raw cost rate for the costed transaction. Oracle Projects uses this information for reporting purposes only.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL. RAW_COST_RATE

INTERFACE_ID

This column is populated by a system–defined value to identify transactions processed by a given concurrent request.

Validation: This column is system assigned.

Destination: None

UNMATCHED_NEGATIVE_TXN_FLAG

A value of Y in this column indicates that the transaction is an unmatched negative transaction.

Enter Y or N so that Transaction Import can identify summary–level adjustments (negative amounts) for which there is no single matching item to adjust. If this column is set to Y, Transaction Import will bypass the matching validation logic that is usually executed for adjustments (negative transactions). If this column is set to N, Oracle Projects finds the matching item and populates PA_EXPENDITURE_ITEMS_ALL.ADJUSTED_EXPENDITURE_ITEM_ID.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.ADJUSTED_EXPENDITURE_ITEM_ID (if column is set to N)

EXPENDITURE_ITEM_ID

This column is populated by a system–defined value to identify the transactions created in Oracle Projects.

Validation: This column is system assigned.

Destination: PA_EXPENDITURES_ALL.EXPENDITURE_ITEM_ID

ORG_ID

This column is populated by the identification code of the organization to which the user belongs. This information is used only if you have implemented multi–organization support.

Validation: Must be a valid organization ID defined in the following tables: PA_EXPENDITURE_GROUP_ALL.ORG_ID, PA_EXPENDITURES_ALL.ORG_ID, PA_EXPENDITURE_ITEMS_ALL.ORG_ID, and PA_COST_DISTRIBUTION_LINES_ALL.ORG_ID.

Destination: None

DR_CODE_COMBINATION_ID

If you are importing a transaction that has already been accounted for and interfaced to GL, enter the ID of the GL debit account. If you allow adjustments to these transactions, Oracle Projects uses this value to account for reversing, adjusting costs.

Validation: Must be a valid GL account in GL_CODE_COMBINATIONS

Destination: PA_COST_DISTRIBUTION_LINES_ALL.DR_CODE_COMBINATION_ID

CR_CODE_COMBINATION_ID

If you are importing a transaction that has already been accounted for and interfaced to GL, enter the ID of the GL credit account.

Validation: Must be a valid GL account in GL_CODE_COMBINATIONS

Destination: PA_COST_DISTRIBUTION_LINES_ALL.
CR_CODE_COMBINATION_ID

CDL_SYSTEM_REFERENCE1

Enter the reference to the record in the external system if it has already been accounted for and interfaced to General Ledger. This information enables you to drill down to the transaction in the originating system.

Validation: None

Destination: PA_COST_DISTRIBUTION_LINES_ALL.
SYSTEM_REFERENCE1

CDL_SYSTEM_REFERENCE2

Enter the reference to the record in the external system if it has already been accounted for and interfaced to General Ledger. This information enables you to drill down to the transaction in the originating system.

Validation: None

Destination: PA_COST_DISTRIBUTION_LINES_ALL.
SYSTEM_REFERENCE2

CDL_SYSTEM_REFERENCE3

Enter the reference to the record in the external system if it has already been accounted for and interfaced to General Ledger. This information enables you to drill down to the transaction in the originating system.

Validation: None

Destination: PA_COST_DISTRIBUTION_LINES_ALL.
SYSTEM_REFERENCE3

GL_DATE

Enter the GL date of the transaction if it has already been accounted for and interfaced to General Ledger. Oracle Projects uses this information for reporting purposes only.

Validation: If this column is null for an accounted transaction, then Transaction Import will reject the transaction.

Destination: PA_COST_DISTRIBUTION_LINES_ALL.
GL_DATE

BURDENED_COST

The system populates this column with the project currency burdened cost for transactions that meet either of the following criteria:

- An expenditure type class of Burden Transaction
- A transaction source with the Import Burdened Amounts option enabled

Burden transactions have quantities and raw costs equal to zero.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.BURDEN_COST

BURDENED_COST_RATE

Enter the burdened cost multiplier for the burden transaction. Oracle Projects uses this information for reporting purposes only.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
BURDEN_COST_RATE

SYSTEM_LINKAGE

Enter the expenditure type class of the given transaction. If the transaction has no expenditure type class, the default expenditure type class defined for the transaction source will be used. Oracle Projects stores this information at the expenditure item level and uses it to determine how to process the expenditure item.

Validation: Must be defined for the expenditure type. If this value is NULL, then the default system linkage (or expenditure type class) defined for the transaction source is used.

Destination: PA_EXPENDITURE_ITEMS_ALL.
SYSTEM_LINKAGE_FUNCTION

TXN_INTERFACE_ID

The values in this column are generated by a sequence to provide a unique identifier for each transaction loaded into the interface table.

Validation: This column is system assigned.

Destination: None

USER_TRANSACTION_SOURCE

Populate this column with the transaction source name. Oracle Projects will populate TRANSACTION_SOURCE based on this value if you do not specify the transaction source code. You can specify either value.

This column is a translatable transaction source column.

Validation: The transaction source you enter must be a valid transaction source. You can obtain a list of valid transaction sources from PA_TRANSACTION_USER.USER_TRANSACTION_SOURCE. Oracle Projects uses values from this table to derive the transaction source if you do not specify a value for the transaction source.

Destination: None

RECEIPT_CURRENCY_AMOUNT

The amount of the expenditure in the original currency (receipt currency).

Validation: If SYSTEM_LINKAGE is Expense Reports, the item is uncosted, and RECEIPT_CURRENCY_CODE is different from DENOM_CURRENCY_CODE, this value must equal zero or null.

Destination: PA_EXPENDITURE_ITEMS_ALL.RECEIPT_CURRENCY_AMOUNT

RECEIPT_CURRENCY_CODE

The currency code for the receipt currency (the currency in which an expense report transaction occurred).

Validation: Must be a valid currency code. A list of valid currency codes can be obtained from FND_CURRENCIES_VL.CURRENCY_CODE and FND_CURRENCIES_VL.ENABLED_FLAG.

Destination: PA_EXPENDITURE_ITEMS_ALL.RECEIPT_CURRENCY_CODE

RECEIPT_EXCHANGE_RATE

The exchange rate to convert from the receipt currency to the transaction (reimbursement) currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
RECEIPT_EXCHANGE_RATE

DENOM_CURRENCY_CODE

The currency code for the transaction currency (reimbursement currency for expense reports).

Validation: Must be a valid currency code. A list of valid currency codes can be obtained from FND_CURRENCIES_VL.CURRENCY_CODE and FND_CURRENCIES_VL.ENABLED_FLAG.

Destination: PA_EXPENDITURE_ITEMS_ALL.
DENOM_CURRENCY_CODE and
PA_COST_DISTRIBUTION_LINES_ALL.
DENOM_CURRENCY_CODE

DENOM_RAW_COST

The raw cost amount in the transaction currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
DENOM_RAW_COST and
PA_COST_DISTRIBUTION_LINES_ALL.
DENOM_RAW_COST

DENOM_BURDENED_COST

The burdened cost amount in the transaction currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
DENOM_BURDENED_COST and
PA_COST_DISTRIBUTION_LINES_ALL.
DENOM_BURDENED_COST

ACCT_RATE_DATE

The exchange rate date for converting to the functional currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL. ACCT_RATE_DATE and PA_COST_DISTRIBUTION_LINES_ALL. ACCT_RATE_DATE

ACCT_RATE_TYPE

The conversion type for converting to the functional currency.

Validation: Must be a valid conversion type. You can obtain a list of valid conversion types from PA_CONVERSION_TYPES_V.

Destination: PA_EXPENDITURE_ITEMS_ALL. ACCT_RATE_TYPE and PA_COST_DISTRIBUTION_LINES_ALL. ACCT_RATE_TYPE

ACCT_EXCHANGE_RATE

The exchange rate for converting to the functional currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL. ACCT_EXCHANGE_RATE and PA_COST_DISTRIBUTION_LINES_ALL. ACCT_EXCHANGE_RATE

ACCT_RAW_COST

The raw cost in the functional currency. For accounted transactions, Transaction Import compares this value to the value calculated from DENOM_RAW_COST, using the conversion attributes. It is validated to make sure that it is within the ACCT_EXCHANGE_ROUNDING_LIMIT. See: Rounding Limit: page 13 – 15.

Validation: For accounted transactions, the functional raw cost, which the system calculates using the given rate attributes (ACCT_RATE_DATE and ACCT_RATE_TYPE) must be within the rounding limit (ACCT_EXCHANGE_ROUNDING_LIMIT) of the entered ACCT_RAW_COST.

Destination: PA_EXPENDITURE_ITEMS_ALL. ACCT_RAW_COST and

PA_COST_DISTRIBUTION_LINES_ALL.
ACCT_RAW_COST

ACCT_BURDENED_COST

The burdened cost in the functional currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
ACCT_BURDENED_COST

ACCT_EXCHANGE_ROUNDING_LIMIT

The functional currency rounding limit. If the derivation of the functional currency raw cost is within the rounding limit, a transaction is accepted. If not, it is rejected. See: Rounding Limit: page 13 – 15.

If the value of ACCT_EXCHANGE_ROUNDING_LIMIT is null, then the rounding limit value used is zero (0).

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
ACCT_ROUNDING_LIMIT

PROJECT_CURRENCY_CODE

This column is derived by the system, based on the project number.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
PROJECT_CURRENCY_CODE and
PA_COST_DISTRIBUTION_LINES_ALL.
PROJECT_CURRENCY_CODE

PROJECT_RATE_DATE

The exchange rate date for converting to the project currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
PROJECT_RATE_DATE and
PA_COST_DISTRIBUTION_LINES_ALL.
PROJECT_RATE_DATE

PROJECT_RATE_TYPE

The conversion rate type for converting to the project currency.

Validation: Must be a valid conversion type. You can obtain a list of valid conversion types from PA_CONVERSION_TYPES_V.

Destination: PA_EXPENDITURE_ITEMS_ALL. PROJECT_RATE_TYPE and PA_COST_DISTRIBUTION_LINES_ALL. PROJECT_RATE_TYPE

PROJECT_EXCHANGE_RATE

The exchange rate for converting to the project currency.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL. PROJECT_EXCHANGE_RATE and PA_COST_DISTRIBUTION_LINES_ALL. PROJECT_EXCHANGE_RATE

ORIG_EXP_TXN_REFERENCE1

Expenditure identifier in the external system (system reference). For supplier invoices imported from Oracle Payables, this column populated by the value of DIST.INVOICE_ID. This column is also used for additional grouping.

Validation: None

Destination: PA_EXPENDITURES_ALL. ORIG_EXP_TXN_REFERENCE1

ORIG_EXP_TXN_REFERENCE2

Columns provided for additional grouping of transactions into expenditures. This column is also used for additional grouping.

Validation: None

Destination: PA_EXPENDITURES_ALL. ORIG_EXP_TXN_REFERENCE2

ORIG_EXP_TXN_REFERENCE3

Columns provided for additional grouping of transactions into expenditures. This column is also used for additional grouping.

Validation: None

Destination: PA_EXPENDITURES_ALL. ORIG_EXP_TXN_REFERENCE3

ORIG_USER_EXP_TXN_REFERENCE

Expenditure identifier in the external system (user reference). For supplier invoices imported from Oracle Payables, this column is populated by the value of INV.INVOICE_NUM.

Validation: None

Destination: PA_EXPENDITURES_ALL. ORIG_USER_EXP_TXN_REFERENCE

VENDOR_NUMBER

The supplier number. For supplier invoices imported from Oracle Payables, this column is populated by the value of INV.VENDOR_ID.

Validation: Must be a valid vendor number (PA_VENDORS.SEGMENT1).

Destination: The corresponding supplier ID is stored in PA_EXPENDITURES_ALL. VENDOR_ID

OVERRIDE_TO_ORGANIZATION_NAME

Override organization name.

Validation: Must be a valid organization name in HR_ORGANIZATION_UNITS.

Destination: The corresponding organization ID is stored in PA_EXPENDITURE_ITEMS_ALL. OVERRIDE_TO_ORGANIZATION_ID

REVERSED_ORIG_TXN_REFERENCE

The reference identifier of the original transaction that this transaction reverses.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
ADJUSTED_EXPENDITURE_ITEM_ID

BILLABLE_FLAG

The billable or capitalizable flag.

Validation: None

Destination: PA_EXPENDITURE_ITEMS_ALL.
BILLABLE_FLAG

PERSON_BUSINESS_GROUP_NAME

Enter the business group name for the person who incurred the charge for this transaction. This column is optional. However, if the employee is defined in more than one business group and this field is not populated, the transaction will be reported as an error.

Validation: Must be a valid organization defined in HR_ALL_ORGANIZATION_UNITS.NAME and be defined in HR_ORGANIZATION_INFORMATION with an OR_INFORMATION_CONTEXT of Business Group Information.

Destination: None

CREATED_BY

This column is populated by the employee number of the user who originally created the expenditure in the Review Transactions window.

Validation: None

Destination: None

CREATION_DATE

This column is populated by the date on which the expenditure was created in the Review Transactions window.

Validation: None

Destination: None

LAST_UPDATED_BY

This column is populated by the employee number of the user who last updated the expenditure in the Review Transactions window.

Validation: None

Destination: None

LAST_UPDATE_DATE

This column is populated by the date on which the expenditure was last updated in the Review Transactions window.

Validation: None

Destination: None

Resolving Import Exceptions

You must correct rejected transactions before you can load them into Oracle Projects. You can correct transaction data in Oracle Projects using the Review Transactions window, or in your external feeder system before you reload the data.

If you correct exceptions in your external system, you must delete the rejected rows from the interface table before reloading the corrected transactions.

This section describes how to correct rejected data, and describes reports you can use to help resolve exceptions.

Examples of Rejection Reason Codes

Transaction Import may reject importing transactions for a variety of reasons. Examples of rejection reasons and their descriptions are shown in the following table:

Rejection Reason	Description
DUPLICATE_ITEM	A transaction with the same transaction source and original transaction reference has already been imported into Oracle Projects (and the transaction source options do not allow duplicate references).
INVALID_END_DATE	The value for the expenditure ending date is not a valid week ending date.
INVALID_PROJECT	No project exists with the project number specified.
ITEM_NOT_IN_WEEK	The expenditure item date for a timecard item does not fall within the timecard expenditure week.
PA_EXP_TASK_TC	The transaction violates an expenditure transaction control at the task level.
PA_EXP_TYPE_INACTIVE	The expenditure item date falls outside the effective dates of the expenditure type. Change the expenditure item date, expenditure type, or expenditure type dates.

Table 13 – 13 Transaction import rejection reasons (Page 1 of 1)

You can get a complete listing of all the rejection reasons from the PA_LOOKUPS table under the lookup type TRANSACTION REJECTION REASON. The codes are also listed in the Oracle eTechnical Reference Manuals (eTRM), in the description of the PA_TRANSACTION_INTERFACE_ALL table. Oracle eTRM is available on *OracleMetaLink*.

Viewing Rejected Transactions

Transaction records that fail the validation process remain in the interface table.

If any one expenditure item in an expenditure fails validation, Oracle Projects rejects the entire expenditure and updates each expenditure item in the expenditure with a status of R (Rejected). However, only rthe expenditure item that was rejected appears on the exception report. Other expenditure items attached to the expenditure being rejected do not appear on the report. Also, the report specifies rejection reasons only for transactions with invalid data. The rest of the expenditures within the batch interface to Oracle Projects. The following tables demonstrate these concepts.

Examples of Transaction Exceptions

The following table shows three transactions before Transaction Import:

Transaction	Status	Reason	Expenditure ID
1	P	[blank]	[blank]
2	P	[blank]	[blank]
3	P	[blank]	[blank]

Table 13 – 14 Before Transaction Import (Page 1 of 1)

The following table shows the same three transactions after Transaction Import. Only Transaction 1 is shown in the exception report.

Transaction	Status	Reason	Expenditure ID
1	P	INVALID PROJECT	1009
2	P	[blank]	1009
3	P	[blank]	1009

Table 13 – 15 AfterTransaction Import (Page 1 of 1)

There are three methods you can use to view rejected transactions:

- Use the Review Transactions window

  You can use the Review Transactions window to search for rejected transactions by transaction source or batch name. See: To view rejected transactions: page 13 – 53.

- Use SQL*Plus

  You can use SQL*Plus to identify the records that have been rejected by selecting those rows with a TRANSACTION_STATUS_CODE of R and selecting the rejection reason for each rejected record from the TRANSACTION_REJECTION_CODE column.

- Review an Oracle Projects report

  The Transaction Import Exception Report shows you all of the transactions that were rejected during the Transaction Import process. For each rejected transaction, this report displays the key field values of the transaction in the interface table. It also displays the rejection reason code that identifies the cause of the

transaction's rejection. See: Transaction Import Report, *Oracle Projects Fundamentals*.

▶ **To view rejected transactions:**

1. In the Navigator window, choose Expenditures > Transaction Import > Review Transactions.
2. Optionally enter the transaction source or the name of the expenditure batch containing the failed transaction(s).

   If you do not enter any search criteria, Oracle Projects will retrieve all rejected transactions, sorted by transaction source and batch name.
3. Choose Find.

Review Transactions Window: Currency–Related Fields

The Review Transactions window is a folder–type window. Many of the currency fields are not displayed in the default folder. You may want to create folders that display the fields you need, for the types of entries you need to make.

For information about folder forms see: Administering Folders, *Oracle Applications System Administrator's Guide*.

Correcting Rejected Transactions within Oracle Projects

If you need to make changes to the source information because of invalid data, you need to delete the rejected rows from the interface table, correct the rejected transactions in the feeder system, and reload them from the feeder system. You can also correct the transaction in the interface table using the Review Transactions window. Oracle Projects automatically updates the status of corrected items and all other transactions in the same expenditure to P (Pending).

The original and updated values for corrected transactions are stored in the audit table PA_TXN_INTERFACE_AUDIT_ALL.

▶ **To correct and resubmit rejected transactions:**

1. After you use the Review Transactions window to query your rejected transactions, make the changes indicated by the transaction rejection reasons. Oracle Projects validates each transaction and displays any errors before proceeding to the next transaction. Acknowledge each error message by choosing OK if

you want to save the transaction with the errors, or choose Cancel and correct the error.

2. Save your work.
3. Choose Import to re–import all the records with a status of Pending for this transaction source and batch. Oracle Projects will validate the transactions online.

You can also use the Review Transactions window to create one or more new transactions without loading them from the feeder system. This window was designed to expedite minor additions to expenditure batches, primarily for testing purposes.

▶ **To create new transactions:**

1. In the Review Transactions window, choose Edit > New Record.
2. Enter transaction details for the new transaction. The information you must enter depends on the transaction source details, just as when you populate the Transaction Import interface table. See: The Transaction Import Interface Table: page 13 – 27.
3. Save your work.
4. Choose Import to start the Transaction Import process.

Correcting Rejected Transactions Using SQL*Plus

You can alternately update the rejected transactions in the interface table using SQL*Plus. Then update the TRANSACTION_ STATUS_CODE column to set the value to P so Transaction Import selects the items the next time you run it. When you resubmit updated transactions for processing, all validation is performed again.

Example: Correcting a Rejected Transaction

Let's walk through an example of the steps you take to correct a rejected transaction using the rejected transaction in the Examples of Transaction Exceptions: page 13 – 52 as our sample data.

1. Correct the invalid data for Transaction 1.

The validation process rejected Transaction 1 because the project you are charging is invalid. Using SQL*Plus, you update the project number of the transaction to a valid project number.

2. Run Transaction Import

Now that you have corrected the rejected expenditure item, and the status of all expenditure items within the rejected expenditure is

updated, you can run Transaction Import to successfully import the updated transactions.

Auditing Updates in the Interface Table

You can update rejected and pending transactions in the interface table using the Review Transactions window or SQL*Plus. Whenever you update a transaction, the original and revised transactions are stored in the PA_TXN_INTERFACE_AUDIT_ALL table. Each transaction is uniquely identified by:

- The combination of the transaction source and original system reference
- The transaction interface ID (if the transaction source allows duplicate system references)

Glossary

account The business relationship that a party can enter into with another party. The account has information about the terms and conditions of doing business with the party.

account combination A unique combination of segment values that records accounting transactions. A typical account combination contains the following segments: company, division, department, account and product.

Account Generator A feature that uses Oracle Workflow to provide various Oracle Applications with the ability to construct Accounting Flexfield combinations automatically using custom construction criteria. You define a group of steps that determine how to fill in your Accounting Flexfield segments. You can define additional processes and / or modify the default process(es), depending on the application. See also *activity, function, item type, lookup type, node, process, protection level, result type, transition, Workflow Engine.*

Account segment One of up to 30 different sections of your Accounting Flexfield, which together make up your general ledger account combination. Each segment typically represents an element of your business structure, such as Company, Cost Center or Account.

Account segment value A series of characters and a description that define a unique value for a particular value set.

account site A party site that is used within the context of an account, for example, for billing or shipping purposes.

accounting currency In some financial contexts, a term used to refer to the currency in which accounting data is maintained. In this manual, this currency is called functional currency. See *functional currency.*

accounting transaction A debit or credit to a general ledger account.

Accounting Flexfield The code you use to identify a general ledger account in an Oracle Financials application. Each Accounting Flexfield segment value corresponds to a summary or rollup account within your chart of accounts.

Accounting Flexfield structure The account structure you define to fit the specific needs of your organization. You choose the number of segments, as well as the length, name, and order of each segment in your Accounting Flexfield structure.

Accounting Flexfield value set A group of values and attributes of the values. For example, the value length and value type that you assign to your account segment to identify a particular element of your business, such as Company, Division, Region, or Product.

accrue through date The date through which you want to accrue revenue for a project. Oracle Projects picks up expenditure items having an expenditure item date on or before this date, and events having a completion date on or before this date, when accruing revenue. An exception to this rule are projects that use cost–to–cost revenue accrual; in this case, the accrue through date used is the PA Date of the expenditure item's cost distribution lines.

accumulation See *summarization.*

activity In Oracle Workflow, a unit of work performed during a business process.

activity In Oracle Receivables, a name that you use to refer to a receivables activity such as a payment, credit memo, or adjustment. See also *activity attribute, function activity, receivables activity name.*

activity attribute A parameter for an Oracle Workflow function activity that controls how the function activity operates. You define an activity attribute by displaying the activity's Attributes properties page in the Activities window of Oracle Workflow Builder. You assign a value to an activity attribute by displaying the activity node's Attribute Values properties page in the Process window.

actual transactions Recorded project costs. Examples include labor, expense report, usage, burden, and miscellaneous costs.

ad hoc For the specific purpose, case, or situation at hand and for no other. For example, an ad hoc tax code, report submission, or database query.

administrative assignment Activity on an administrative project such as personal holiday, sick day, or jury duty. Administrative assignments can also represent administrative work such as duties on an internal project. Such assignments are charged to the administrative project which is determined by the administration flag on the project type.

advance An amount of money prepaid in anticipation of receipt of goods, services, obligations or expenditures.

advance In Oracle Payables, an advance is a prepayment paid to an employee. You can apply an advance to an employee expense report during expense report entry, once you fully pay the advance.

agreement A contract with a customer that serves as the basis for work authorization. An agreement may represent a legally binding contract, such as a purchase order, or a verbal authorization. An agreement sets the terms of payment for invoices generated against the agreement, and affects whether there are limits to the amount of revenue you can accrue or bill against the agreement. An agreement can fund the work of one or more projects.

agreement type An implementation–defined classification of agreements. Typical agreement types include purchase order and service agreement.

allocation A method for distributing existing amounts between and within projects and tasks. The allocation feature uses existing project amounts to generate expenditure items for specified projects.

allocation method An attribute of an allocation rule that specifies how the rule collects and allocates the amounts in the source pool. There are two allocation methods, full allocation and incremental allocation. See also *full allocation*, *incremental allocation*.

allocation rule A set of attributes that describes how you want to allocate amounts in a source pool to specified target projects and tasks.

allocation run The results of the PRC: Generate Allocation Transactions process.

alternative region An alternative region is one of a collection of regions that occupy the same space in a window where only one region can be displayed at any time. You identify an alternative region by a poplist icon that displays the region title, which sits on top of a horizontal line that spans the region. This display method has been replaced by tabs in Release 11i and higher.

amount class For allocations, the period or periods during which the source pool accumulates amounts.

amount type The starting point for a time interval. Available options include period–to–date, year–to–date, and project–to–date. Used to define budgetary controls for a project.

analysis workbook A display of enterprise information in a graphical and tabular format. The Analysis Workbook uses Discoverer to enable the user to modify the selection criteria, drill into dimension hierarchies, or link to other data elements.

approved date The date on which an invoice is approved.

archive To store historical transaction data outside your database.

asset An object of value owned by a corporation or business. Assets are entered in Oracle Projects as non–labor resources. See *non–labor resource.* See *fixed asset*.

assignment forecast item Assignment Forecast Item is the smallest unit of forecasting information for the assignment. In this entity, the smallest time unit is a day. Forecast items are created for each day of every provisional and confirmed assignment for every billable resource.

attribute See *activity attribute, item type attribute.*

attribute In TCA, corresponds to a column in a TCA registry table, and the attribute value is the value that is stored in the column. For example, party name is an attribute and the actual values of party names are stored in a column in the HZ_PARTIES table.

AutoAccounting In Oracle Projects, a feature that automatically determines the account coding for an accounting transaction based on the project, task, employee, and expenditure information.

AutoAccounting In Oracle Receivables, a feature that lets you determine how the Accounting Flexfields for your revenue, receivable, freight, tax, unbilled receivable and unearned revenue account types are created.

AutoAccounting function A group of related AutoAccounting transactions. There is at least one AutoAccounting function for each Oracle Projects process that uses AutoAccounting. AutoAccounting functions are predefined by Oracle Projects.

AutoAccounting Lookup Set An implementation–defined list of intermediate values and corresponding Accounting Flexfield segment values. AutoAccounting lookup sets are used to translate intermediate values such as organization names into account codes.

AutoAccounting parameter A variable that is passed into AutoAccounting. AutoAccounting parameters are used by AutoAccounting to determine account codings. Example AutoAccounting parameters available for an expenditure item are the expenditure type and project organization. AutoAccounting parameters are predefined by Oracle Projects.

AutoAccounting Rule An implementation–defined formula for deriving Accounting Flexfield segment values. AutoAccounting rules may use a combination of AutoAccounting parameters, AutoAccounting lookup sets, SQL statements, and constants to determine segment values.

AutoAccounting Transaction A repository of the account coding rules needed to create one accounting transaction. For each accounting transaction created by Oracle Projects, the necessary AutoAccounting rules are held in a corresponding AutoAccounting Transaction. AutoAccounting transactions are predefined by Oracle Projects.

autoallocation set A group of allocation rules that you can run in sequence that you specify (step–down allocations) or at the same time (parallel allocations). See also *step–down allocation***,** *parallel allocation.*

AutoInvoice A program that imports invoices, credit memos, and on–account credits from other systems to Oracle Receivables.

automatic event An event with an event type classification of Automatic. Billing extensions create automatic events to account for the revenue and invoice amounts calculated by the billing extensions.

AutoReduction An Oracle Applications feature in the list window that allows you to shorten a list so that you must scan only a subset of values before choosing a final value. Just as AutoReduction incrementally reduces a list of values as you enter additional character(s), pressing [Backspace] incrementally expands a list.

AutoSelection A feature in the list window that allows you to choose a valid value from the list with a single keystroke. When you display the list window, you can type the first character of the choice you want in the window. If only one choice begins with the character you enter, AutoSelection selects the choice, closes the list window, and enters the value in the appropriate field.

AutoSkip A feature specific to flexfields where Oracle Applications automatically moves your cursor to the next segment as soon as you enter a valid value into a current flexfield segment. You can turn this feature on or off with the user profile option Flexfields:AutoSkip.

availability Availability of a resource for a specified duration is presented in the form of a percentage calculated as follows:

(capacity minus the number of confirmed assignments hours)100 / capacity

availability match See availability

balancing segment An Accounting Flexfield segment that you define so that General Ledger automatically balances all journal entries for each value of this segment. For example, if your company segment is a balancing segment, General Ledger ensures that, within every journal entry, the total debits to company 01 equal the total credits to company 01

baseline To approve a budget for use in reporting and accounting.

baseline budget The authorized budget for a project or task which is used for performance reporting and revenue calculation.

basis method How an allocation rule is used to allocate the amounts from a source pool to target projects. The basis methods include options to spread the amounts evenly, allocate by percentage, or prorate amounts based on criteria you specify. Also referred to as the ”basis.” See also *source pool*.

batch source A source you define in Oracle Receivables to identify where your invoicing activity originates. The batch source also controls invoice defaults and invoice numbering. Also known as a **transaction batch source**.

bill rate A rate per unit at which an item accrues revenue and / or is invoiced for time and material projects. Employees, jobs, expenditure types, and non–labor resources can have bill rates.

bill rate schedule A set of standard bill rates that maintains the rates and percentage markups over cost that you charge clients for your labor and non–labor expenditures.

bill site The customer address to which project invoices are sent.

bill through date The date through which you want to invoice a project. Oracle Projects picks up revenue distributed expenditure items having an expenditure item date on or before this date, and events having a completion date on or before this date, when generating an invoice.

billable resource A resource that has a current billable job assignment. Billable jobs are defined in the job definition screen where the Job Billability flag is set to Y.

billing The functions of revenue accrual and invoicing.

billing cycle The billing period for a project. Examples of billing cycles you can define are: a set number of days, the same day each week or month, or the project completion date. You can optionally use a client extension to define a billing cycle.

billing title See *Employee Billing Title*, *Job Billing Title.*

block Every Oracle Applications window (except root and modal windows) consists of one or more blocks. A block contains information pertaining to a specific business entity Generally, the first or only block in a window assumes the name of the window. Otherwise, a block name appears across the top of the block with a horizontal line marking the beginning of the block.

borrowed and lent A method of processing cross charge transactions that generates accounting entries to pass cost or share revenue between the provider and receiver organizations within a legal entity. See also: Intercompany Billing.

boundary code The end point for a time interval. Available options include period, year, and project. Used to define budgetary controls for a project.

budget Estimated cost, revenue, labor hours or other quantities for a project or task. Each budget may optionally be categorized by resource. Different budget types may be set up to classify budgets for different purposes. In addition, different versions can exist for each user–defined budget type: current, original, revised original, and historical versions. The current version of a budget is the most recently baseline version. See also *budget line*, *resource.*

budgetary controls Control settings that enable the system to monitor and control project–related commitment transactions.

budget line Estimated cost, revenue, labor hours, or other quantity for a project or task categorized by a resource.

burden cost code An implementation–defined classification of overhead costs. A burden cost code represents the type of burden cost you want to apply to raw cost. For example, you can define a burden cost code of G&A to burden specific types of raw costs with General and Administrative overhead costs.

burden costs Burden costs are legitimate costs of doing business that support raw costs and cannot be directly attributed to work performed. Examples of burden costs are fringe benefits, office space, and general and administrative costs.

burden multiplier A numeric multiplier associated with an organization for burden schedule revisions, or with burden cost codes for projects or tasks. This multiplier is applied to raw cost to calculate burden cost amounts. For example, you can assign a multiplier of 95% to the burden cost code of Overhead.

burden schedule An implementation–defined set of burden multipliers that is maintained for use across projects. Also referred to as a *standard burden schedule*. You may define one or more schedules for different purposes of costing, revenue accrual, and invoicing. Oracle Projects applies the burden multipliers to the raw cost amount of an expenditure item to derive an amount; this amount may be the total cost, revenue amount, or bill amount. You can override burden schedules by entering negotiated rates at the project and task level. See also *Firm Schedule, Provisional Schedule, Burden Schedule Revision*, *Burden Schedule Override.*

burden schedule override A schedule of negotiated burden multipliers for projects and tasks that overrides the schedule you defined during implementation.

burden schedule revision A revision of a set of burden multipliers. A schedule can be made of many revisions.

burden structure A burden structure determines how cost bases are grouped and what types of burden costs are applied to the cost bases. A burden structure defines relationships between cost bases and burden cost codes and between cost bases and expenditure types.

burdened cost The cost of an expenditure item, including raw cost and burden costs.

business entity A person, place, or thing that is tracked by your business. For example, a business entity can be an account, a customer, or a part.

business group The highest level of organization and the largest grouping of employees across which a company can report. A business group can correspond to an entire company, or to a specific division within the company. Each installation of Oracle Projects uses one business group with one hierarchy.

business view Component of the application database that sorts underlying applications data into an understandable and consolidated set of information. By masking the complexity of the database tables, Business Views provide a standard set of interfaces to any tool or application that retrieves and presents data to the user.

button You choose a button to initiate a predefined action. Buttons do not store values. A button is usually labeled with text to describe its action or it can be an icon whose image illustrates its action.

calendar Working capacity defined by work patterns and calendar exceptions.

capacity Capacity is the total number of hours a resource can be scheduled based on the calendar of the resource. In the case of Labor, capacity is defined in work hours. The capacity of an Organization is the sum total of the capacity of assigned resources.

capital project A project in which you build one or more depreciable fixed assets.

chart of accounts The account structure your organization uses to record transactions and maintain account balances.

chart of accounts structure See: *Accounting Flexfield Structure: page Glossary – 2*.

check box You can indicate an on / off or yes / no state for a value by checking or unchecking its check box. One or more check boxes can be checked since each check box is independent of other check boxes.

child request A concurrent request submitted by another concurrent request (a parent request.) For example, each of the reports and / or programs in a report set are child requests of that report set.

CIP assets See: *construction–in–process assets.*

chargeable project For each expenditure, a project to which the expenditure can be charged or transferred.

claim A discrepancy between the billed amount and the paid amount. Claims are often referred to as deductions, but a claim can be positive or negative.

class category An implementation–defined category for classifying projects. For example, if you want to know the market sector to which a project belongs, you can define a class category with a name such as *Market Sector.* Each class category has a set of values (class codes) that can be chosen for a project. See *class code.*

class code An implementation–defined value within a class category that can be used to classify a project. See *class category.*

clearing account An account used to ensure that both sides of an accounting transaction are recorded. For example, Oracle General Ledger uses clearing accounts to balance intercompany transactions.

When you purchase an asset, your payables group creates a journal entry to the asset clearing account. When your fixed assets group records the asset, they create an offset journal entry to the asset clearing account to balance the entry from the payables group.

combination block A combination block displays the fields of a record in both multi–record (summary) and single–record (detail) formats. Each format appears in its own separate window that you can easily navigate between.

combination query See *Existing Combinations.*

comment alias A user–defined name for a frequently used line of comment text, which can be used to facilitate online entry of timecards and expense reports.

commitment transactions Anticipated project costs. Examples include purchase requisitions and purchase orders, provisional and confirmed contract commitments, and supplier invoices.

competence A technical skill or personal ability such as JAVA programming, customer relations, and project billing.

competence match A numerical comparison of the competence of a resource to the mandatory and optional competencies of a requirement. In the candidate score calculation, this number is converted to a percentage.

complete matching A condition where the invoice quantity matches the quantity originally ordered, and you approve the entire quantity. See also *matching, partial matching.*

construction–in–process (CIP) asset A depreciable fixed asset you plan to build during a capital project. The costs associated with building CIP assets are referred to as CIP costs. See also *capital project*. You construct CIP assets over a period of time rather than buying a finished asset**.** Oracle Assets lets you create, maintain, and add to your CIP assets as you spend money for material and labor to construct them. When you finish the assets and place them in service (capitalize them), Oracle Assets begins depreciating them.

concurrent manager A unique facility that manages many time–consuming, non–interactive tasks within Oracle Applications. When you submit a request that does not require your interaction, such as releasing shipments or running a report, the Concurrent Manager does the work for you, letting you complete multiple tasks simultaneously.

concurrent process A non–interactive task that you request Oracle Applications to complete. Each time you submit a non–interactive task, you create a new concurrent process. A concurrent process runs simultaneously with other concurrent processes (and other interactive activities on your computer) to help you complete multiple tasks at once.

concurrent queue A list of concurrent requests awaiting completion by a concurrent manager. Each concurrent manager has a queue of requests waiting to be run. If your system administrator sets up your Oracle Application to have simultaneous queuing, your request can wait to run in more than one queue.

concurrent request A request to Oracle Applications to complete a non–interactive task for you, such as releasing a shipment, posting a journal entry, or running a report. Once you submit a request, Oracle Applications automatically completes your request.

contact In Oracle Projects, a customer representative who is involved with a project. For example, a contact can be a billing contact, the customer representative who receives project invoices.

contact point A means of contacting a party other than postal mail, for example, a phone number, e–mail address, fax number, and so on.

contact type An implementation–defined classification of project contacts according to their role in the project. Typical contact types are Billing and Shipping.

context field prompt A question or prompt to which a user enters a response, called a context field value. When Oracle Applications displays a descriptive flexfield pop–up window, it displays your context field prompt after it displays any global segments you have defined. Each descriptive flexfield can have up to one context prompt.

context field value A response to your context field prompt. Your response is composed of a series of characters and a description. The response and description together provide a unique value for your context prompt, such as 1500, Journal Batch ID, or 2000, Budget Formula Batch ID. The context field value determines which additional descriptive flexfield segments appear.

context response See *context field value*.

context segment value A response to your context–sensitive segment. The response is composed of a series of characters and a description. The response and description together provide a unique value for your context–sensitive segment, such as Redwood Shores, Oracle Corporation Headquarters, or Minneapolis, Merrill Aviation's Hub.

context–sensitive segment A descriptive flexfield segment that appears in a second pop–up window when you enter a response to your context field prompt. For each context response, you can define multiple context segments, and you control the sequence of the context segments in the second pop–up window. Each context–sensitive segment typically prompts you for one item of information related to your context response.

contract project A project for which you can generate revenue and invoices. Typical contract project types include Time and Materials and Fixed Price. Formerly known as a **direct project**.

control level The level of control to impose on project transactions during a funds check. Available options are absolute, advisory, and none. Used to define budgetary controls for a project.

controlled budget A budget for which budgetary controls have been enabled.

conversion A process that converts foreign currency transactions to your functional currency. See also *foreign currency conversion*.

corporate exchange rate An exchange rate you can optionally use to perform foreign currency conversion. The corporate exchange rate is usually a standard market rate determined by senior financial management for use throughout the organization. You define this rate in Oracle General Ledger.

cost base A cost base refers to the grouping of raw costs to which burden costs are applied. Examples of cost bases are Labor and Materials.

cost budget The estimated cost amounts at completion of a project. Cost budget amounts can be summary or detail, and can be burdened or unburdened.

cost burden schedule A burden schedule used for costing to derive the total cost amount. You assign the cost burden schedule to a project type that is burdened; this default cost burden schedule defaults to projects that use the project type; and then from the project to the tasks below the project. You may override the cost burden schedule for a project or a task if you have defined the project type option to allow overrides of the cost burden schedule.

cost distribution The act of calculating the cost and determining the cost accounting for an expenditure item.

cost rate The monetary cost per unit of an employee, expenditure type, or resource.

cost–to–cost A revenue accrual method that calculates project revenue as budgeted revenue multiplied by the ratio of actual cost to budgeted cost. Also known as **percentage of completion method or percentage spent method**.

credit memo In Oracle Payables and Oracle Projects, a document that partially or fully reverses an original invoice.

In Oracle Receivables, a document that partially or fully reverses an original invoice. You can create credit memos in the Receivables Credit Transactions window or with AutoInvoice.

Cross Business Group Access (CBGA) The ability to view data in operating units that are not associated with the current operating unit's business group.

Cross Business Group Access mode (CBGA mode) An installation that has selected CBGA in the profile options is operating in **CBGA mode.**

cross charge To charge a resource to a project owned by a different operating unit.

credit receiver A person receiving credit for project or task revenue. One project or task may have many credit receivers for one or many credit types.

credit type An implementation–defined classification of the credit received by a person for revenue a project earns. Typical credit types include Quota Credit and Marketing Credit.

Cross–Project responsibility A responsibility that permits users to view and update any project.

cross charge transaction An expenditure item whose provider operating unit is different from the receiver operating unit, the provider organization is different from the receiver organization, or both.

cross charge project A project that can receive transactions from an operating unit or organization that is different from the operating unit or organization that owns the project.

cross charge type One of the three types of cross charge transactions: intercompany, inter–operating unit, and intra–operating unit.

cross–project user A user who is logged into Oracle Projects using a Cross–Project responsibility.

current budget The most recently baseline budget version of the budget.

current record indicator Multi–record blocks often display a current record indicator to the left of each record. A current record indicator is a one character field that when filled in, identifies a record as being currently selected.

customer agreement See *agreement*.

database table A basic data storage structure in a relational database management system. A table consists of one or more units of information (rows), each of which contains the same kind of values (columns). Your application's programs and windows access the information in the tables for you.

deferred revenue An event type classification that generates an invoice for the amount of the event, and has no immediate effect on revenue. The invoice amount is accounted for in an unearned revenue account that will be offset as the project accrues revenue.

delivery assignment Filled work position on a project that is not an administrative project.

denomination currency In some financial contexts, a term used to refer to the currency in which a transaction takes place. In this manual, this currency is called transaction currency. See: *transaction currency.*

depreciate To depreciate an asset is to spread its cost over the time you use it. You charge depreciation expense for the asset each period. The total depreciation taken for an asset is stored in the accumulated depreciation account.

Descriptive Flexfield A field that your organization can extend to capture extra information not otherwise tracked by Oracle Applications. A descriptive flexfield appears in your window as a single character, unnamed field. Your organization can customize this field to capture additional information unique to your business.

direct project An obsolete term. See *contract project*.

dimension An Oracle Financial Analyzer database object used to organize and index the data stored in a variable. Dimensions are used in Oracle Project to calculate and monitor performance measures. Dimensions answer the following questions about data: "What?" "When?" and "Where?" For example, a variable called Units Sold might be associated with the dimensions Product, Month, and District. In this case, Units Sold describes the number of products sold during specific months within specific districts.

distribution line In Oracle Payables and Oracle Projects, a line corresponding to an accounting transaction for an expenditure item on an invoice, or a liability on a payment.

distribution line In Oracle Assets, information such as employee, general ledger depreciation expense account, and location to which you have assigned an asset. You can create any number of distribution lines for each asset. Oracle Assets uses distribution lines to allocate depreciation expense and to produce your Property Tax and Responsibility Reports.

distribution rule See *revenue distribution rule.*

draft budget A preliminary budget which may be changed without affecting revenue accrual on a project.

draft invoice A potential project invoice that is created, adjusted, and stored in Oracle Projects. Draft invoices require approval before they are officially accounted for in other Oracle Applications.

draft revenue A project revenue transaction that is created, adjusted, and stored in Oracle Projects. You can adjust draft revenue before you transfer it to other Oracle Applications.

drilldown A software feature that allows you to view the details of an item in the current window via a window in a different application.

duration The total number of days between the start date and end date of a team role.

dynamic insertion An optional Accounting Flexfields feature that allows you to create new account combinations during data entry in Oracle Applications. By enabling this feature, it prevents having to define every possible account combination that can exist. Define cross–validation rules when using this feature.

effort The total number of hours of a team role.

employee billing title An employee title, which differs from a job billing title, that may appear on an invoice. Each employee can have a unique employee billing title.

employee organization The organization to which an employee is assigned.

encumbrance A journal entry to reserve funds for anticipated project costs (commitments). The primary purpose for posting encumbrances is to avoid overspending a budget.

End User Layer Component of Discoverer that translates business view column names into industry standard terminology and provides links between related data tables. Discoverer accesses information through the End User Layer (EUL).

euro A single currency adopted by the member states of the European Union. The official abbreviation, EUR, is used for all commercial, business, and financial purposes, and has been registered with the International Standards Organization (ISO).

event In Oracle Projects, a summary level transaction assigned to a project or top task that records work completed and generates revenue and / or billing activity, but is not directly related to any expenditure items. For example, unlike labor costs or other billable expenses, a bonus your business receives for completing a project ahead of schedule is not attributable to any expenditure item, and would be entered as an event.

event type An implementation–defined classification of events that determines the revenue and invoice effect of an event. Typical event types include Milestones, Scheduled Payments, and Write–Offs.

exchange rate A rate that represents the amount one currency can be exchanged for another at a specific point in time. Oracle Applications can access daily, periodic, and historical rates. These rates are used for foreign currency conversion, revaluation, and translation.

exchange rate type The source of an exchange rate. For example, user defined, spot, or corporate rate. See also: corporate exchange: page Glossary – 10 rate, spot exchange rate: page Glossary – 34.

Existing Combinations A feature specific to key flexfields in data entry mode that allows you to enter query criteria in the flexfield to bring up a list of matching predefined combinations of segment values to select from.

expenditure A group of expenditure items incurred by an employee or an organization for an expenditure period. Typical expenditures include Timecards and Expense Reports.

expenditure (week) ending date The last day of an expenditure week period. All expenditure items associated with an expenditure must be on or before the expenditure ending date, and must fall within the expenditure week identified by the expenditure week ending date.

expenditure category An implementation–defined grouping of expenditure types by type of cost. For example, an expenditure category with a name such as *Labor* refers to the cost of labor.

expenditure comment Free text that can be entered for any expenditure item to explain or describe it in further detail.

expenditure cost rate The monetary cost per unit of a non–labor expenditure type.

expenditure cycle A weekly period for grouping and entering expenditures.

expenditure group A user–defined name used to track a group of pre–approved expenditures, such as Timecards, or Expense Reports.

expenditure item The smallest logical unit of expenditure you can charge to a project and task. For example, an expenditure item can be a timecard item or an expense report item.

expenditure item date The date on which work is performed and is charged to a project and task.

expenditure operating unit The operating unit in which an expenditure is entered and processed for project costing.

expenditure organization For timecards and expense reports, the organization to which the incurring employee is assigned, unless overridden by organization overrides. For usage, supplier invoices, and purchasing commitments, the incurring organization entered on the expenditure.

expenditure type An implementation–defined classification of cost that you assign to each expenditure item. Expenditure types are grouped into cost groups (expenditure categories) and revenue groups (revenue categories).

expenditure type class An additional classification for expenditure types that indicates how Oracle Projects processes the expenditure types. For example, if you run the Distribute Labor Costs process, Oracle Projects will calculate the cost of all expenditure items assigned to the Straight Time expenditure type class. Formerly known as **system linkage**.

expense report In Oracle Payables, a document that details expenses incurred by an employee for the purpose of reimbursement. You can enter expense reports online in Payables, or employees enter them online in Internet Expenses. You can then submit Expense Report Import to import these expense reports and expense reports from Projects. The import program creates invoices in Payables from the expense report data.

expense report In Oracle Projects, a document that, for purposes of reimbursement, details expenses incurred by an employee. You can set up expense report templates to match the format of your expense reports to speed data entry. You must create invoices from Payables expense reports using Expense Report Import before you can pay the expense reports.

Expense Report Import An Oracle Payables process you use to create invoices from Payables expense reports. You can also use Expense Report Import to create invoices from expense reports in Oracle Projects.

When you initiate Expense Report Import, Payables imports the expense report information and automatically creates invoices with invoice distribution lines from the information. Payables also produces a report for all expense reports it could not import.

external organization See *organization*.

feeder program A custom program you write to transfer your transaction information from an original system into Oracle Application interface tables. The type of feeder program you write depends on the environment from which you are importing data.

field A position on a window that you use to enter, view, update, or delete information. A field prompt describes each field by telling you what kind of information appears in the field, or alternatively, what kind of information you should enter in the field.

firm schedule A burden schedule of burden multipliers that will not change over time. This is compared to provisional schedules in which actual multipliers are mapped to provisional multipliers after an audit.

first bill offset days The number of days that elapse between a project start date and the date that the project's first invoice is issued.

fixed asset An item owned by your business and used for operations. Fixed assets generally have a life of more than one year, are acquired for use in the operation of the business, and are not intended for resale to customers. Assets differ from inventory items since you use them rather than sell them**.**

fixed date See *schedule fixed date*.

flat file A file where the data is unformatted for a specific application.

flexfield An Oracle Applications field made up of segments. Each segment has an assigned name and a set of valid values. Oracle Applications uses flexfields to capture information about your organization. There are two types of flexfields: key flexfields and descriptive flexfields.

flexfield segment One of the sections of your key flexfield, separated from the other sections by a symbol that you define (such as –, /, or \). Each segment typically represents an element of your business, such as cost center, product, or account.

folder Customizable windows located throughout Oracle Applications. Folders allow you to: change the display of a window by resizing or reordering columns, hide or display columns, and change field names to best fit the needs of each user's working style.

foreign currency In Oracle Applications, a currency that is different from the functional currency you defined for your set of books in Oracle General Ledger. When you enter and pay a foreign currency invoice, Payables automatically converts the foreign currency into your functional currency at the rate you define. General Ledger automatically converts foreign currency journal entries into your functional currency at the rate you define. See also *exchange rate*, *functional currency*.

foreign currency conversion A process in Oracle Applications that converts a foreign currency transaction into your functional currency using and exchange rate you specify.

form A window that contains a logical collection of fields, regions, and blocks that appear on a single screen. You enter data into forms. See *window*.

full allocation An allocation method that distributes all the amounts in the specified projects in the specified amount class. The full allocation method is generally suitable if you want to process an allocation rule only once in a run period. See also *incremental allocation*.

function A PL/SQL stored procedure referenced by an Oracle Workflow function activity that can enforce business rules, perform automated tasks within an application, or retrieve application information. The stored procedure accepts standard arguments and returns a completion result. See also *function activity*.

function activity An automated Oracle Workflow unit of work that is defined by a PL/SQL stored procedure. See also *function*.

function security An Oracle Applications feature that lets you control user access to certain functions and windows. By default, access to functionality is *not* restricted; your system administrator customizes each responsibility at your site by including or excluding functions and menus in the Responsibilities window.

functional currency The principal currency you use to record transactions and maintain accounting data for your set of books. Also, in cross charge transactions, the currency, as defined in the set of books, associated with a project transaction. For example, the cost functional currency is the functional currency for both the project expenditure item and the set of books of the expenditure operating unit. The invoice functional currency is the functional currency for both the project revenue and the set of books of the project operating unit.

funds check The process that verifies a budget's available funds. When budgetary controls are enabled, a funds check is performed against the project budget for commitment transactions. When top–down budgeting is also enabled, a funds check is performed against the funding budget for the project budget lines.

GL Date The date, referenced from Oracle General Ledger, used to determine the correct accounting period for your transactions.

In Oracle Projects, the end date of the GL Period in which costs or revenue are transferred to Oracle General Ledger. This date is determined from the open or future GL Period on or after the Project Accounting Date of a cost distribution line or revenue. For invoices, the date within the GL Period on which an invoice is transferred to Oracle Receivables.

global hierarchy An organization hierarchy that includes one or more business groups. A global hierarchy can be used by installations that are in CBGA mode.

global security profile An HR security profiles that is not associated with a business group. A global security profile can secure organizations and people throughout a global (cross business group) organization hierarchy.

global segment prompt A non–context–sensitive descriptive flexfield segment. Each global segment typically prompts you for one item of information related to the zone or form in which you are working.

global segment value A response to your global segment prompt. Your response is composed of a series of characters and a description. The response and description together provide a unique value for your global segment, such as J. Smith, Financial Analyst, or 210, Building C.

hard limit An option for an agreement that prevents revenue accrual and invoice generation beyond the amount allocated to a project or task by the agreement. If you do not impose a hard limit, Oracle Projects automatically imposes a soft limit of the same amount. See also *soft limit*.

HR job In HRMS, the HR job for a resource (person) is the job linked to the primary assignment of the person.

incremental allocation An allocation method that creates expenditure items based on the difference between the transactions processed from one allocation to the next. This method is generally suitable if you want to use an allocation rule in allocation runs several times in a given run period. See also *full allocation*.

indirect project A project used to collect and track costs for overhead activities, such as administrative labor, marketing, and bid and proposal preparation. You can also define indirect projects to track time off such as sick leave, vacation, and holidays. You cannot generate revenue or invoices for indirect projects.

inter–operating unit cross charge transaction An expenditure item for which the provider and receiver operating units are different, but both operating units are associated with the same legal entity.

intercompany billing A method of internally billing work performed by a provider operating unit and charged to a project owned by a receiver operating unit. The provider operating unit creates a Receivables invoice, which is interfaced as a Payables invoice to the receiver operating unit. See: *Borrowed and Len*t.

intercompany billing project A contract project set up in the provider operating unit to process intercompany billing. The provider operating unit must create one intercompany billing project for each receiver operating unit it wants to charge.

intercompany cross charge transaction An expenditure item that crosses legal entity boundaries, which means that the provider and receiver operating units are different and are associated with different legal entities.

intercompany invoice base amount The sum of the amounts in the provider's transfer price functional currency.

intercompany invoice currency The transaction currency of an intercompany invoice. You can specify the invoice currency attributes for each intercompany billing project to convert the intercompany invoice base amount to the intercompany invoice amount

intermediate value The parameter value, constant, or SQL statement result that is determined during the first step in the execution of an AutoAccounting rule.

internal billing Intercompany billing for work performed between two organizations or projects. The process creates the appropriate documents so the provider operating unit can bill the receiver operating unit.

internal organization See *organization*.

internal requisition See *internal sales order, purchase requisition*.

internal sales order A request within your company for goods or services. An internal sales order originates from an employee or from another process as a requisition, such as inventory or manufacturing, and becomes an internal sales order when the information is transferred from Purchasing to Order Management. Also known as **internal requisition** or **purchase requisition.**

intra–operating unit cross charge transaction An cross charge expenditure item charged entirely within an operating unit. The provider and receiver organizations are different, but the provider and receiver operating units are the same.

invoice In Oracle Receivables and Oracle Cash Management, a document that you create in Receivables that lists amounts owed for the purchases of goods or services. This document also lists any tax, freight charges, and payment terms.

invoice In Oracle Payables and Oracle Assets, a document you receive from a supplier that lists amounts owed to the supplier for purchased goods or services. In Payables, you create an invoice online using the information your supplier provides on the document, or you import an invoice from a supplier. Payments, inquiries, adjustments and any other transactions relating to a supplier's invoice are based upon the invoice information you enter.

invoice In Oracle Projects, a summarized list of charges, including payment terms, invoice item information, and other information that is sent to a customer for payment.

invoice burden schedule A burden schedule used for invoicing to derive the bill amount of an expenditure item. This schedule may be different from your revenue burden schedule, if you want to invoice at a different rate at which you want to accrue.

invoice currency The currency in which an Oracle Projects invoice is issued.

invoice date In Oracle Assets and Oracle Projects, the date that appears on a customer invoice. This date is used to calculate the invoice due date, according to the customer's payment terms.

In Oracle Receivables, the date an invoice is created. This is also the date that Receivables prints on each invoice. Receivables also uses this date to determine the payment due date based on the payment terms you specify on the invoice.

In Oracle Payables, the date you assign to an invoice you enter in Payables. Payables uses this date to calculate the invoice due date, according to the payment terms for the invoice. The invoice date can be the date the invoice was entered or it can be a different date you specify.

invoice distribution line A line representing an expenditure item on an invoice. A single expenditure item may have multiple distribution lines for cost and revenue. An invoice distribution line holds an amount, account code, and accounting date.

invoice format The columns, text, and layout of invoice lines on an invoice.

invoice item A single line of a project's draft invoice, formatted according to the project invoice formats.

invoice set For each given run of invoice generation for a project, if multiple agreements exist and multiple invoices are created, Oracle Projects creates the invoices within a unique set ID. You approve, release, and cancel all invoices within an invoice set.

invoice transaction type An Oracle Receivables transaction type that is assigned to invoices and credit memos that are created from Oracle Projects draft invoices.

invoice write–off A transaction that reduces the amount outstanding on an invoice by a given amount and credits a bad debt account.

invoicing The function of preparing a client invoice. Invoice generation refers to the function of creating the invoice. Invoicing is broader in the terms of creating, adjusting, and approving an invoice.

item type A term used by Oracle Workflow to refer to a grouping of all items of a particular category that share the same set of item attributes, used as a high level grouping for processes. For example, each Account Generator item type (e.g. FA Account Generator) contains a group of processes for determining how an Accounting Flexfield code combination is created. See also *item type attribute.*

item type attribute A feature of a particular Oracle Workflow item type, also known as an item attribute. An item type attribute is defined as a variable whose value can be looked up and set by the application that maintains the item. An item type attribute and its value is available to all activities in a process.

Item Validation Organization The organization that contains your master list of items. You must define all items and bills in your Item Validation Organization, but you also need to maintain your items and bills in separate organizations if you want to ship them from other warehouses. Oracle Order Management refers to organizations as warehouses on all Order Management forms and reports. See also *organization*.

job A name for a set of duties to which an employee may be assigned. You create jobs in Oracle Projects by combining a job level and a job discipline using your job key flexfield structure. For example, you can combine the job level *Staff* with the job discipline *Engineer* to create the job *Staff Engineer.*

job billing title A job billing title, which differs from a job title, that may appear on an invoice.

job discipline A categorization of job vocation, used with Job Level to create a job title. For example, a job discipline may be Engineer, or Consultant.

job group A collection of jobs defined for a specific purpose. Jobs in a job group have the same key flexfield structure.

job level A categorization of job rank, used with Job Discipline to create a job title. For example, a job level may be Staff, or Principal.

job level In Oracle Project Resource Management it is a numeric value associated to the job of the Project Resource Job Group. Each resource has a job and an associated job level that either belongs to or is mapped to the Project Resource Job Group. The level provides a basis for searching for potential resource matches. See **job level match.**

job level match A numeric value of 0% or 100%. If the job level of the resource is within the range of specified job levels for the search, then the job level match for the resource is 100, otherwise, it is 0. This percentage is used by the calculation for determining the candidate score.

job title In Oracle Projects, a unique combination of job level and job discipline that identifies a particular job.

job title In Oracle Receivables, a brief description of your customer contact's role within their organization.

journal entry category A category to indicate the purpose or nature of a journal entry, such as Adjustment or Addition. Oracle General Ledger associates each of your journal entry headers with a journal category. You can use one of General Ledger's pre–defined journal categories or define your own.

For Oracle Payables, there are three journal entry categories in Oracle Projects if you use the accrual basis accounting method: Invoices, Payments, and All (both Invoices and Payments). If you use the cash basis accounting method, Oracle Projects only assigns the Payment journal entry category to your journal entries.

journal entry header A method used to group journal entries by currency and journal entry category within a journal entry batch. When you initiate the transfer of invoices or payments to your general ledger for posting, Oracle Payables transfers the necessary information to create journal entry headers for the information you transfer. Journal Import in General Ledger uses the information to create a journal entry header for each currency and journal entry category in a journal entry batch. A journal entry batch can have multiple journal entry headers.

journal entry lines Each journal entry header contains one or more journal entry lines. The lines are the actual journal entries that your general ledger posts to update account balances. The number and type of lines in a journal entry header depend on the volume of transactions, frequency of transfer from Oracle Payables, and your method of summarizing journal entries from Oracle Payables.

journal entry source Identifies the origin of journal entries from Oracle and non–Oracle feeder systems. General Ledger supplies predefined journal sources or you can create your own.

Journal Import A General Ledger program that creates journal entries from transaction data stored in the General Ledger GL_INTERFACE table. Journal entries are created and stored in GL_JE_BATCHES, GL_JE_HEADERS, and GL_JE_LINES.

key flexfield An intelligent key that uniquely identifies an application entity. Each key flexfield segment has a name you assign, and a set of valid values you specify. Each value has a meaning you also specify. You use this Oracle Applications feature to build custom fields used for entering and displaying information relating to your business. The following application uses the listed Key Flexfields:

Oracle Projects – Accounting, Category Flexfield, Location, Asset Key.

key flexfield segment One of up to 30 different sections of your key flexfield. You separate segments from each other by a symbol you choose (such as –, / or \.). Each segment can be up to 25 characters long. Each key flexfield segment typically captures one element of your business or operations structure, such as company, division, region, or product for the Accounting Flexfield and item, version number, or color code for the Item Flexfield.

key flexfield segment value A series of characters and a description that provide a unique value for this element, such as 0100, Eastern region, or V20, Version 2.0.

key member An employee who is assigned a role on a project. A project key member can view and update project information and expenditure details for any project to which they are assigned. Typical key member types include Project Manager and Project Coordinator.

labor cost The cost of labor expenditure items.

labor cost rate The hourly raw cost rate for an employee. This cost rate does not include overhead or premium costs.

labor costing rule An implementation–defined name for an employee costing method. Also known as pay type. Typical labor costing rules include *Hourly* and *Exempt*.

labor invoice burden schedule A burden schedule used to derive invoice amounts for labor items.

labor multiplier A multiplier that is assigned to a project or task, and is used to calculate the revenue and / or bill amount for labor items by applying the multiplier to the raw cost of the labor items.

labor revenue burden schedule A burden schedule used to derive revenue amounts for labor items.

legal entity An organization that represents a legal company for which you prepare fiscal or tax reports. You assign tax identifiers and other relevant information to this entity.

lifecycle A collection of sequential project phases.

liquidation The process of relieving an encumbrance.

listing An organized display of Oracle Applications information, similar to a report, but usually showing setup data as opposed to transaction data.

Logical Data Model A representation of the End User Layer. Available in a readable format, the Logical Data Model gives the relationship between folders, allowing a Discoverer user to determine the data elements needed for a specific analysis.

lookup code The internal name of a value defined in an Oracle Workflow lookup type. See also *lookup type.*

lookup type An Oracle Workflow predefined list of values. Each value in a lookup type has an internal and a display name. See also *lookup code.*

lowest task A task that has no child tasks.

master–detail relationship A master–detail relationship is an association between two blocks—a master block and its detail block. When two blocks are linked by a master–detail relationship, the detail block displays only those records that are associated with the current (master) record in the master block, and querying between the two blocks is always coordinated. Master and detail blocks can often appear in the same window or they can each appear in separate windows.

master job A job in a master job group.

master job group The job group that is used as an intermediate mapping group between other job groups.

match rule A set of rules that determines which records are matches for an input record. A match rule consists of an acquisition portion to determine potential matches, a scoring portion to score the potential matches, and thresholds that the scores are compared against to determine actual matches.

matching In Oracle Cash Management, the process where batches or detailed transactions are associated with a statement line based on the transaction number, amount, currency and other variables, taking Cash Management system parameters into consideration. In Cash Management, matching can be done manually or automatically.

matching In Oracle Payables and Oracle Assets, the process of comparing purchase order, invoice, and receiving information to verify that ordering, billing, and receiving information is consistent within accepted tolerance levels. Payables uses matching to control payments to suppliers. You can use the matching feature in Payables if you have Purchasing or another purchasing system. Payables supports two–, three–, and four–way matching.

message line A line on the bottom of a window that displays helpful hints or warning messages when you encounter an error.

mid task A task that is not a top task or a lowest task.

multi–org See *multiple organizations.*

multiple organizations The ability to define multiple organizations and the relationships among them within a single installation of Oracle Applications. These organizations can be sets of books, business groups, legal entities, operating units, or inventory organizations.

Multiple Reporting Currencies A unique set of features embedded in Oracle Applications that allows you to maintain and report accounting records at the transaction level in more than one functional currency.

node An instance of an activity in an Oracle Workflow process diagram as shown in the Process window of Oracle Workflow Builder. See also *process.*

non–capacity work type Work types assigned to forecast assignment items or actual expenditure items reduce the total capacity of a given resource for the specified time period.

non–invoice related claim A claim that is due to a discrepancy between the billed amount and the paid amount, and cannot be identified with a particular transaction.

non–labor invoice burden schedule A burden schedule used to derive invoice amounts for non–labor items.

non–labor resource An implementation–defined asset or pool of assets. For example, you can define a non–labor resource with a name such as *PC* to represent multiple personal computers your business owns.

non–labor revenue burden schedule A burden schedule used to derive revenue amounts for non–labor items.

non–project budget A budget defined outside Oracle Projects. Examples include organization–level budgets defined in Oracle General Ledger, and budgets defined in Oracle Contract Commitments.

non–revenue sales credit Sales credit you assign to your salespeople that is not associated with your invoice lines. This is sales credit given in excess of your revenue sales credit. See also *revenue sales credit*.

offsets Reversing transactions used to balance allocation transactions with the source or other project.

one time billing hold A type of hold that places expenditure items and events on billing hold for a particular invoice; when you release that invoice, the items are billed on the next invoice.

operating unit An organization that partitions data for subledger products (AP, AR, PA, PO, OE). It is roughly equivalent to a single pre–Multi–Org installation.

operator A mathematical symbol you use to indicate the mathematical operation in your calculation.

option group An option group is a set of option buttons. You can choose only one option button in an option group at a time, and the option group takes on that button's value after you choose it. An option button or option group is also referred to as a radio button or radio group, respectively.

Oracle Discoverer An Oracle tool that enables users to retrieve data from a database. Oracle Discoverer provides a user friendly method for creating database queries and displaying information.

organization A business unit such as a company, division, or department. Organization can refer to a complete company, or to divisions within a company. Typically, you define an organization or a similar term as part of your account when you implement Oracle Financials. See also *business group*.

Internal organizations are divisions, groups, cost centers or other organizational units in a company. External organizations can include the contractors your company employs. Organizations can be used to demonstrate ownership or management of functions such as projects and tasks, non–labor resources, and bill rate schedules. See also *Item Validation Organization*.

organization hierarchy An organizational hierarchy illustrates the relationships between your organizations. A hierarchy determines which organizations are subordinate to other organizations. The topmost organization of an organization hierarchy is generally the business group.

organization structure See *organization hierarchy.*

original budget The budget amounts for a project at the first successful baseline of the project.

Overtime Calculation Program A program that Oracle Projects provides to determine which kind of overtime to award an employee based on the employee's labor costing rule and hours worked. If your company uses this automatic overtime calculation feature, you may need to modify the program based on the overtime requirements of your business.

overtime cost The currency amount over straight time cost that an employee is paid for overtime hours worked. Also referred to as Premium Cost.

PA Date The end date of the PA Period in which costs are distributed, revenue is created, or an invoice is generated. This date is determined from the open or future PA Period on or after the latest date of expenditure item dates and event completion dates included in a cost distribution line, revenue, or an invoice.

PA Period See *Project Accounting Period.*

PA Period Type The Period Type as specified in the PA implementation options for Oracle Projects to copy project accounting periods. Oracle Projects uses the periods in the PA Period Type to populate each Operating Unit's PA periods. PA periods are mapped to GL periods which are used when generating accounting transactions. PA periods drive the project summary for Project Status Inquiry. You define your accounting periods in the Operating Unit's Set of Books Calendar.

parallel allocation A set of allocation rules that carries out the rules in an autoallocation set without regard to the outcome of the other rules in the set. See also *autoallocation set, step–down allocation*.

parent request A concurrent request that submits other concurrent requests (child requests). For example, a report set is a parent request that submits reports and / or programs (child requests).

partial matching A condition where the invoice quantity is less than the quantity originally ordered, in which case you are matching only part of a purchase order shipment line. See also *matching*, *complete matching*.

pay type See *labor costing rule*.

phase A collection of logically related project activities, usually culminating in the completion of a major deliverable.

pop–up window An additional window that appears on an Oracle Applications form when your cursor enters a particular field.

poplist A poplist, when selected by your mouse, lets you choose a single value from a predefined list.

posting The process of updating account balances in Oracle General Ledger from journal entries. Payables uses the term posting to describe the process of transferring accounting entries to General Ledger. Payables transfers your invoice and payment accounting entries and sets the status of the payments and invoices to posted. You must then complete the process by creating and posting the journal entries in General Ledger. Note that Oracle Applications sometimes use the term posting to describe the process of transferring posting information to your general ledger. See also *Journal Import*.

premium cost See *overtime cost*.

primary contact A person in the organization with resource authority.

primary set of books The set of books you use to manage your business. You can choose accrual or cash basis as the accounting method for your primary set of books.

process A set of Oracle Workflow activities that need to be performed to accomplish a business goal. See also *Account Generator*, *process activity*, *process definition*.

process activity An Oracle Workflow process modelled as an activity so that it can be referenced by other processes; also known as a subprocess. See also *process*.

process cycle The planned schedule for batch processing of costs, revenue, and invoices, according to your company's scheduling requirements. See *streamline request*.

process definition An Oracle Workflow process as defined in the Oracle Workflow Builder. See also *process*.

process responsibility type An implementation–defined name to which a group of reports and processes are assigned. This group of reports and processes is then assigned to an Oracle Projects responsibility. A process responsibility type gives a user access to Oracle Projects reports and programs appropriate to that user's job. For example, the process responsibility type Data Entry could be a set of reports used by data entry clerks. See *responsibility*.

product lifecycle management A process for guiding products from their birth through their completion. The lifecycle management process adds business value to an enterprise by using product information to support planning, monitoring, and execution of vital activities.

profile option A set of options that control access to certain features throughout Oracle Applications and determines how data is processed. Generally, profile options can be set at the Site, Application, Responsibility, and User levels. For more information, see the user guide for your specific Oracle Application.

project A unit of work that can be broken down into one or more tasks. A project is the unit of work for which you specify revenue and billing methods, invoice formats, a managing organization and project manager, and bill rate schedules. You can charge costs to a project, and you can generate and maintain revenue, invoice, unbilled receivable, and unearned revenue information for a project.

Project Accounting Period An implementation–defined period against which project performance may be measured. Also referred to as *PA Periods*. You define project accounting periods to track project accounting data on a periodic basis by assigning a start date, end date, and closing status to each period. Typically, you define project accounting periods on a weekly basis, and your general ledger periods on a monthly basis.

Project Burdening Organization Hierarchy The organization hierarchy version that Oracle Projects uses to compile burden schedules. Each business group must designate one and only one version of an organization hierarchy as its Project Burdening Organization Hierarchy. (Note: In Oracle Projects Implementation Options, each operating unit is associated with an organization hierarchy and version for project setup, invoice level processing, and project reporting. The Project Burdening Organization Hierarchy selected for the business group does not have to match the hierarchy version in the Implementation Options.).

project chargeable employees In a multiple organization installation, employees included as labor resource pool to a project. This includes all employees, as defined in Oracle Human Resources, who belong to the business group associated with the project operating unit.

project currency The currency in which project transactions are billed (unless overridden during the billing process). Also, the currency in which project amounts are summarized for project summary reporting.

project funding An allocation of revenue from an agreement to a project or task.

project operating unit The operating unit within which the project is created, and in which the project customer revenue and receivable invoices are processed.

project resource group The job group used to identify appropriate roles for use within Project Resource Management.

project/task organization The Organization that owns the project or task. This can be any organization in the LOV (list of values) for the project setup. The Project / Task Organization LOV contains organizations of the Project / Task Organization Type in the Organization Hierarchy and Version below the Start Organization. You specify your Start Organization and Version in the Implementation Options window.

project role An implementation–defined classification of the relationship that an employee has to a project. You use project roles to define an employee's level of access to project information.

project status An implementation–defined classification of the status of a project. Typical project statuses are Active and Closed.

project template A standard project you create for use in creating other projects. You set up project templates that have features common in the projects you want to create.

project type A template defined for your implementation. The template consists of project attributes such as the project type class (contract, indirect, or capital), the default revenue distribution rule and bill rate schedules, and whether the project burdens costs. For example, you can define a project type with a name such as *Time and Materials* for all projects that are based on time and materials contracts.

project type class An additional classification for project types that indicates how to collect and track costs, quantities, and, in some cases, revenue and billing. Oracle Projects predefines three project type classes: *Indirect*, *Contract*, or *Capital*. For example, you use an Indirect project type to collect and track project costs for overhead activities, such as administrative and overhead work, marketing, and bid and proposal preparation.

Project/customer relationship An implementation–defined classification of the relationship between a project and a customer. Project / Customer Relationships help you manage projects that involve multiple clients by specifying the various relationships your customers can have with a project. Typical relationships include Primary or Non–Paying.

Project/Task Alias A user–defined short name for a project or project / task combination used to facilitate online timecard and expense report entry.

Project/Task Organization The Organization that owns the project or task.

protection level In Oracle Workflow, a numeric value ranging from 0 to 1000 that represents who the data is protected from for modification. When workflow data is defined, it can either be set to customizable (1000), meaning anyone can modify it, or it can be assigned a protection level that is equal to the access level of the user defining the data. In the latter case, only users operating at an access level equal to or lower than the data's protection level can modify the data. See also *Account Generator*.

provider operating unit The operating unit whose resources provide services to another project or organization. For cross charge transactions, the provider operating unit is the expenditure operating unit; the project operating unit owns the intercompany billing project.

provider organization For cross charge transactions, the organization that provides resources to another organization. The default is the expenditure organization or the non–labor resource organization, which can be overridden using the Provider and Receiver Organization Override client extension.

provider project The contract project that performs work on behalf of another (receiver) project.

provider transfer price functional currency The functional currency of the set of books for the *provider operating unit*.

provider transfer price functional currency amount The currency amount calculated by applying the transfer price currency conversion attributes (as specified by the implementation options for the provider operating unit) to the transfer price base currency amount.

provisional schedule A burden schedule of estimated burden multipliers that are later audited to determine the actual rates. You apply actual rates to provisional schedules by replacing the provisional multipliers with actual multipliers. Oracle Projects processes adjustments that account for the difference between the provisional and actual calculations.

purchase order (PO) In Oracle General Ledger and Oracle Projects, a document used to buy and request delivery of goods or services from a supplier.

purchase order (PO) In Oracle Assets, the order on which the purchasing department approved a purchase.

purchase order distribution Each purchase order shipment consists of one or more purchase order distributions. A purchase order distribution consists of the Accounting Flexfield information Payables uses to create invoice distributions.

purchase order line An order for a specific quantity of a particular item at a negotiated price. Each purchase order in Purchasing can consist of one or more purchase order lines.

purchase order requisition line Each purchase order line is created from one or more purchase order requisition lines. Purchasing creates purchase order requisition lines from individual requisitions.

purchase requisition An internal request for goods or services. A requisition can originate from an employee or from another process, such as inventory or manufacturing. Each requisition can include many lines, generally with a distinct item on each requisition line. Each requisition line includes at least a description of the item, the unit of measure, the quantity needed, the price per item, and the Accounting Flexfield you are charging for the item. Also known as **internal requisition**. See also *internal sales order.*

purchasing site A supplier site from which you order goods or services. You must enter at least one purchasing site before Purchasing will allow you to enter a purchase order.

query A search for applications information that you initiate using an Oracle Applications window.

raw costs Costs that are directly attributable to work performed. Examples of raw costs are salaries and travel expenses.

receipt currency The currency in which an expense report item originates.

record A record is one occurrence of data stored in all the fields of a block. A record is also referred to as a row or a transaction, since one record corresponds to one row of data in a database table or one database transaction.

receiver operating unit An operating unit whose projects receive services from another project or organization. For inter–project billing , the receiver operating unit is the project operating unit that owns the receiver project.

receiver organization The operating unit whose projects receive services from another project or organization. For cross charged transactions, the receiver operating unit is the project operating unit that owns

receiver project A project for which work is performed by another (provider) project. In inter–project billing, the receiver project incurs costs from a Payables invoice generated by the Receivables tieback process performed by the provider project.

receiver task A task in the receiver project to which costs are assigned on the Payables invoice.

region A collection of logically–related fields set apart from other fields by a dashed line that spans a block. Regions help to organize a block so that it is easier to understand. Regions in Release 11i and higher are defined by Tabs.

reimbursement currency The currency in which an employee chooses to be reimbursed for an expense report. See also *transaction currency.*

related transaction Additional transactions that are created for labor transactions using the Labor Transaction Extension. All related transactions are associated with a *source transaction* and are attached to the expenditure item ID of the source transaction. You can identify and process the related transactions by referring to the expenditure item ID of the source transaction. Using labor transaction extensions, you can create, identify, and process the related transactions along with the source transaction.

released date The date on which an invoice and its associated revenue is released.

remit to addresses The address to which your customers remit their payments.

report an organized display of information drawn from Oracle Applications that can be viewed online or printed. Most applications provide standard and customizable reports. Oracle General Ledger's Financial Statement Generator lets you build detailed financial reports and statements based on your business needs.

report headings A descriptive section found at the top of each report detailing general information about the report such as set of books, date, etc.

report option See *report parameter*.

report parameter Submission options in Oracle Applications that allow you to enter date and account ranges. You can also sort, format, select, and summarize the information displayed in your reports. Most standard reports require you enter report parameters.

report security group A feature that helps your system administrator control your access to reports and programs. Your system administrator defines a report security group which consists of a group of reports and / or programs and assigns a report security group to each responsibility that has access to run reports using Standard Report Submission. When you submit reports using Standard Report Submission, you can only choose from those reports and programs in the report security group assigned to your responsibility.

report set A group of reports that you submit at the same time to run as one transaction. A report set allows you to submit the same set of reports regularly without having to specify each report individually. For example, you can define a report set that prints all of your regular month–end management reports.

requirement Unfilled work position on a project.

resource A user–defined group of employees, organizations, jobs, suppliers, expenditure categories, revenue categories, expenditure types, or event types for purposes of defining budgets or summarizing actuals.

responsibility A level of authority set up by your system administrator in Oracle Applications. A responsibility lets you access a specific set of windows, menus, set of books, reports, and data in an Oracle application. Several users can share the same responsibility, and a single user can have multiple responsibilities.

responsibility type See *process responsibility type*.

result code In Oracle Workflow, the internal name of a result value, as defined by the result type. See also *result type, result value.*

result type In Oracle Workflow, the name of the lookup type that contains an activity's possible result values. See also *result code, result value.*

result value In Oracle Workflow, the value returned by a completed activity, such as *Approved*. See also *result code, result type.*

revenue In Oracle Projects, the amounts recognized as income or expected billing to be received for work on a project.

revenue accrual The function of calculating and distributing revenue.

revenue authorization rule A configurable criterion that, if enabled, must be met before a project can accrue revenue. For example, an active mandatory revenue authorization rule states that a project manager must exist on a project before that project can accrue revenue. Revenue authorization rules are associated with revenue distribution rules. See also *revenue distribution rule*.

revenue budget The estimated revenue amounts at completion of a project. Revenue budget amounts can be summary or detail.

revenue burden schedule A burden schedule used for revenue accrual to derive the revenue amount for an expenditure item. This schedule may be different from your invoice burden schedule, if you want to accrue revenue at a different rate than you want to invoice.

revenue category An implementation–defined grouping of expenditure types by type of revenue. For example, a revenue category with a name such as *Labor* refers to labor revenue.

revenue credit Credit that an employee receives for project revenue. See *revenue sales credit*.

revenue distribution rule A specific combination of revenue accrual and invoicing methods that determine how Oracle Projects generates revenue and invoice amounts for a project. See *revenue authorization rule*.

revenue item A single line of a project's revenue, containing event or expenditure item revenue summarized by top task and revenue category or event.

revenue sales credit Sales credit you assign to your salespeople that is based on your invoice lines. The total percentage of all revenue sales credit must be equal to 100% of your invoice lines amount. Also known as **quota sales credits**. See also *non–revenue sales credit, sales credit.*

revenue write–off An event type classification that reduces revenue by the amount of the write–off. You cannot write–off an amount that exceeds the current unbilled receivables balance on a project. See also *invoice write–off*.

root window The root window displays the main menu bar and tool bar for every session of Oracle Applications. In Microsoft Windows, the root window is titled "Oracle Applications" and contains all the Oracle Applications windows you run. In the Motif environment, the root window is titled "Toolbar" because it displays just the toolbar and main menu bar.

row One occurrence of the information displayed in the fields of a block. A block may show only one row of information at a time, or it may display several rows of information at once, depending on its layout. The term "row" is synonymous with the term "record".

sales credit Credits that you assign to your salespeople when you enter orders, invoices, and commitments. Credits can be either quota or non–quota and can be used in determining commissions. See also *non–revenue sales credit, revenue sales credit*.

sales tax A tax collected by a tax authority on purchases of goods and services. The supplier of the good or service collects sales taxes from its customers (tax is usually included in the invoice amount) and remits them to a tax authority. Tax is usually charged as a percentage of the price of the good or service. The percentage rate usually varies by authority and sometimes by category of product. Sales taxes are expenses to the buyer of goods and services.

salesperson A person who is responsible for the sale of products or services. Salespeople are associated with orders, returns, invoices, commitments, and customers. You can also assign sales credits to your salespeople.

schedule The working hours defined by the calendar and schedule exceptions.

schedule fixed date The date used to freeze bill rate or burden schedules for a project or task. You enter a fixed date to specify that you want to use particular rates or multipliers as of that date. You do not use schedule fixed dates if you want to use the current effective rates or multipliers for a particular schedule.

scrollable region A region whose contents are not entirely visible in a window. A scrollable region contains a horizontal or vertical scroll bar so that you can scroll horizontally or vertically to view additional fields hidden in the region.

segment A single sub–field within a flexfield. You define the structure and meaning of individual segments when customizing a flexfield.

service type An implementation–defined classification of the type of work performed on a task.

set of books Defined in Oracle General Ledger, an organization or group of organizations that share a common chart of accounts, calendar, and currency. A set of books is associated with one or more responsibilities.

To use Multiple Reporting Currencies, you must create a primary set of books and separate reporting sets of books for each reporting currency.

soft limit The default option for an agreement that generates a warning when you accrue revenue or generate invoices beyond the amount allocated to a project or task by the agreement, but does not prevent you from running these processes. See also *hard limit*.

shorthand flexfield entry A quick way to enter key flexfield data using shorthand aliases (names) that represent valid flexfield combinations or patterns of valid segment values. Your organization can specify flexfields that will use shorthand flexfield entry and define shorthand aliases for these flexfields that represent complete or partial sets of key flexfield segment values.

shorthand window A single–segment customizable field that appears in a pop–up window when you enter a key flexfield. The shorthand flexfield pop–up window only appears if you enable shorthand entry for that particular key flexfield.

sign–on An Oracle Applications user name and password that allows you to gain access to Oracle Applications. Each sign–on is assigned one or more responsibilities.

Single Business Group Access mode (SBGA mode) An installation that has selected No for the profile option **HR: Cross Business Group** is operating in **SBGA mode.**

source pool The combination of all the source amounts defined by an allocation rule. See also *allocation rule*.

source transaction For related transactions, the identifying source transaction from which the related items are created.

spot exchange rate A daily exchange rate you use to perform foreign currency conversions. The spot exchange rate is usually a quoted market rate that applies to the immediate delivery of one currency for another.

standard bill rate schedule currency The functional currency of the operating unit in which the standard bill rate schedule is maintained.

Standard Request Submission A standard interface in Oracle Applications in which you run and monitor your application's reports and other processes.

start organization An organization that defines a set which includes itself and all subordinate organizations in the organization hierarchy. When you choose a start organization as a report parameter, all organizations below the start organization are included in the report.

status line A status line appearing below the message line of a root window that displays status information about the current window or field. A status line can contain the following: **^** or **v** symbols indicate previous records before or additional records following the current record in the current block; **Enter Query** indicates that the current block is in Enter Query mode, so you can specify search criteria for a query; **Count** indicates how many records were retrieved or displayed by a query (this number increases with each new record you access but does not decrease when you return to a prior record); the **<Insert>** indicator or *lamp* informs you that the current window is in insert character mode; and the **<List>** lamp appears when a list of values is available for the current field.

step–down allocation In Oracle Projects, a set of allocation rules that carries out the rules (steps) an autoallocation set serially, in the sequence specified in the set. Usually the result of each step will be used in the next step. Oracle Workflow controls the flow of the autoallocations set. See also *autoallocation set*, *parallel allocation*.

straight time cost The monetary amount that an employee is paid for straight time (regular) hours worked.

streamline process See *streamline request*.

streamline request A process that runs multiple Oracle Projects processes in sequence. When using streamline processing, you can reschedule your streamline requests by setting rescheduling parameters. Rescheduling parameters allow you to configure your processes to run automatically, according to a defined schedule. When you reschedule a process, the concurrent manager submits another concurrent request with a status of *Pending*, and with a start date according to the parameters you define.

structure A structure is a specific combination of segments for a key flexfield. If you add or remove segments, or rearrange the order of segments in a key flexfield, you get a different structure.

subtask A hierarchical unit of work. Subtasks are any tasks that you create under a parent task. Child subtasks constitute the lowest level of your work breakdown structure; where Oracle Projects looks when processing task charges and for determining task revenue accrual amounts. See *task*.

summarization Processing a project's cost, revenue, commitment, and budget information to be displayed in the Project, Task, and Resource Project Status windows. You must distribute costs for any expenditure items, accrue and release any revenue, create any commitments, and baseline a budget for your project before you can view summary project amounts. Formerly known as **accumulation**.

supplier A business or individual that provides goods or services or both in return for payment.

supplier invoice An external supplier's invoice entered into Oracle Payables.

system linkage An obsolete term. See *expenditure type class.*

tablespace The area in which an Oracle database is divided to hold tables.

target A project, task, or both that receives allocation amounts, as specified by an allocation rule. See also *source pool*

task A subdivision of project work. Each project can have a set of top level tasks and a hierarchy of subtasks below each top level task. See also *Work Breakdown Structure, subtask.*

task organization The organization that is assigned to manage the work on a task.

task service type See *service type.*

tax authority A governmental entity that collects taxes on goods and services purchased by a customer from a supplier. In some countries, there are many authorities (e.g. state, local, and federal governments in the U.S.), while in others there may be only one. Each authority may charge a different tax rate. You can define a unique tax name for each tax authority. If you have only one tax authority, you can define a unique tax name for each tax rate that it charges. A governmental entity that collects taxes on goods and services purchased by a customer from a supplier. In some countries, there are many authorities (e.g. state, local and federal governments in the U.S.), while in others there may be only one. Each authority may charge a different tax rate. Within Oracle Receivables, tax authority consists of all components of your tax structure. For example: California. San Mateo. Redwood Shores for State. County. City. Oracle Receivables adds together the tax rates for all of these locations to determine a customer's total tax liability for an invoice.

tax codes Codes to which you assign sales tax or value–added tax rates, tax type, taxable basis, tax controls, and tax accounting. You can define a tax code for inclusive or exclusive tax calculation. Oracle Receivables lets you choose state codes as the tax code when you define sales tax rates for the United States. (Receivables Lookup)

team role Specific position on a project representing either requirements or assignments.

Time and Materials (T&M) A revenue accrual and billing method that calculates revenue and billings as the sum of the amounts from each individual expenditure item. The expenditure item amounts are calculated by applying a rate or markup to each item.

time intervals The units that define how budget amounts are accumulated to determine the available funds for a transaction. Used to define budgetary controls for a project.

timecard A weekly submission of labor expenditure items. You can enter timecards online, or as part of a pre–approved batch.

toolbar The toolbar is a collection of iconic buttons that each perform a specific action when you choose it. Each toolbar button replicates a commonly–used menu item. Depending on the context of the current field or window, a toolbar button can be enabled or disabled. You can display a hint for an enabled toolbar button on the message line by holding your mouse steadily over the button. The toolbar generally appears below the main menu bar in the root window.

top task A task whose parent is the project.

transaction currency The currency in which a transaction originally takes place. For processing purposes, the reimbursement currency in an expense report is the transaction currency.

transfer price The price agreed upon by the provider and receiver organizations in a cross charged transaction.

transfer price base currency The transfer price basis determines the currency. For a basis of raw or burdened cost, the transfer price base currency is the transaction currency of the cross charged transaction. For a basis of revenue, the transfer price base currency is the functional currency of the set of books for the receiver operating unit. For a basis calculated using the bill rate schedule, the transfer price base currency is the standard bill rate schedule currency.

transferred date The date on which you transfer costs, revenue, and invoices to other Oracle Applications.

transformation function A seeded or user–defined rule that transforms and standardizes TCA attribute values into representations that can assist in the identification of potential matches.

transition In Oracle Workflow, the relationship that defines the completion of one activity and the activation of another activity within a process. In a process diagram, the arrow drawn between two activities represents a transition. See also *activity, Workflow Engine.*

unassigned time The net amount of hours for a given period for which a resource does not have any scheduled assignments (capacity hours minus scheduled hours.)

unbilled receivables The amount of open receivables that have not yet been billed for a project. Oracle Projects calculates unbilled receivables using the following formula: *(Unbilled Receivables = Revenue Accrued – Amount Invoice)*

unearned revenue Revenue received and recorded as a liability or revenue before the revenue has been earned by providing goods or services to a customer. Oracle Projects calculates unearned revenue using the following formula: *(Unearned Revenue = Amount Invoiced – Revenue Accrued)*

unit of measure A classification created in Oracle General Ledger that you assign to transactions in General Ledger and subledger applications. Each unit of measure belongs to a unit of measure class.

For example, if you specify the unit of measure Miles when you define an expenditure type for personal car use, Oracle Projects calculates the cost of using a personal car by mileage. Or, in Oracle Payables, you define square feet as a unit of measure. When you enter invoices for office rent, you can track the square footage addition to the dollar amount of the invoice.

In Oracle Assets, a label for the production quantities for a units of production asset. The unit used to measure production amounts.

UOM See *unit of measure.*

usage See *non–labor resource*.

usage cost rate override The cost rate assigned to a particular non–labor resource and non–labor organization which overrides the rate assigned to its expenditure type.

usage logs Usage logs record the utilization of company assets on projects as the asset is used.

user profile A set of changeable options that affect the way your applications run. You can change the value of a user profile option at any time. See *profile option.*

utilization A measure of how effectively a resource was used or is projected to be used.

utilization method *Capacity Utilization Method* compares the actual (productive) work performed and forecasted (productive) work to be performed by the resource to the capacity of a resource.

Worked Hours Utilization Method compares the actual (productive) work performed and forecasted (productive) work to be performed by the resource to the total number of hours recorded (actuals) or assigned (forecasted) of a resource.

utilization category An implementation–defined category used for utilization reporting. This reporting grouping combines one or more work types for organization and resource utilization views.

utilization view Utilization views enable you to measure a resource or organization utilization percentage based on different groupings of work types.

value Data you enter in a parameter. A value can be a date, a name, or a code, depending on the parameter.

value set A group of values and related attributes you assign to an account segment or to a descriptive flexfield segment. Values in each value set have the same maximum length, validation type, alphanumeric option, and so on.

vendor See *supplier.*

window A box around a set of related information on your screen. Many windows can appear on your screen simultaneously and can overlap or appear adjacent to each other. Windows can also appear embedded in other windows. You can move a window to a different location on your screen.

window title A window title at the top of each window indicates the name of the window, and occasionally, context information pertinent to the content of the window. The context information, contained in parenthesis, can include the organization, set of books, or business group that the window contents is associated with.

WIP See *work in process.*

word replacement A word mapping that is used to create synonyms which are treated as equivalents for searching and matching.

work breakdown structure (WBS) The breakdown of project work into tasks. These tasks can be broken down further into subtasks, or hierarchical units of work.

work in process An item in various phases of production in a manufacturing plant. This includes raw material awaiting processing up to final assemblies ready to be received into inventory.

work site The customer site where project or task work is performed.

work type Work types are an implementation–defined classification of work performed. Work types are used to classify both actual and forecast amounts. Examples are Billable, Non–Billable, Training, and Personal. Work types are grouped together by Utilization Categories.

worksheet A specific grouping of information within an Analysis Workbook. A workbook is composed of one or more worksheets, each with its own set of data and graphs. Conceptually, this is similar to the ‘‘sheets” and ‘‘workbook” concept within a spreadsheet application.

Workflow Engine The Oracle Workflow component that implements a workflow process definition. The Workflow Engine manages the state of all activities, automatically executes functions, maintains a history of completed activities, and detects error conditions and starts error processes. The Workflow Engine is implemented in server PL/SQL and activated when a call to an engine API is made. See also *Account Generator, activity, function, item type.*

write–off See *invoice write–off, revenue write–off.*

write–on An event type classification that causes revenue to accrue and generates an invoice for the amount of the write–on.

Zoom A forms feature that is obsolete in GUI versions of Oracle Applications.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Reader's Comment Form

Oracle Projects APIs, Client Extensions, and Open Interfaces Reference Release 11*i*

Part Number B12427–01

Oracle 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?

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

- Electronic mail: appsdoc_us@oracle.com
- Fax: (650) 506–7200 Attn: Oracle Projects
- Postal Service

  Oracle Applications Documentation Manager
  Oracle Corporation
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA
  Phone: (650) 506–7000

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

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