The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS
Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee’s responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
# Contents

## Send Us Your Comments

## Preface

### Part 1  OVERVIEW

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

- Overview of Oracle Projects APIs, Client Extensions, and Open Interfaces................................................. 1-1
  - Oracle Projects APIs.................................................................................................................................. 1-1
  - Oracle Projects Client Extensions............................................................................................................. 1-2
  - Oracle Projects Open Interfaces.................................................................................................................. 1-2

### Part 2  ORACLE PROJECTS APIS

2  **Introduction to Oracle Projects APIs**

- Introduction to Oracle Projects APIs............................................................................................................ 2-1
- Overview of the Oracle Projects APIs............................................................................................................. 2-2
  - Applications of the Oracle Projects APIs.................................................................................................. 2-2
  - Where Information Originates..................................................................................................................... 2-3
- Integrating Your External System with Oracle Projects.................................................................................. 2-5
  - Step 1: Create a Database Role.................................................................................................................. 2-5
  - Step 2: Create an Oracle Applications User................................................................................................. 2-6
  - Step 3: Create a Database User................................................................................................................... 2-6
  - Step 4: Set Up Your Product in Oracle Projects.......................................................................................... 2-7
- Security Requirements....................................................................................................................................... 2-8
3 Oracle Project Foundation APIs

Project Definition APIs................................................................. 3-2
  Project Definition API Views.................................................... 3-2
  Project Definition API Procedures............................................ 3-6
Project Definition API Procedure Definitions............................ 3-7
  Common Project Definition API Parameters.......................... 3-7
  CREATE_PROJECT........................................................................ 3-8
  DELETE_PROJECT........................................................................ 3-9
  UPDATE_PROJECT........................................................................ 3-10
  CLEAR_PROJECT......................................................................... 3-17
  EXECUTE_CREATE_PROJECT...................................................... 3-17
  EXECUTE_UPDATE_PROJECT....................................................... 3-18
  INIT_PROJECT............................................................................ 3-19
  LOAD_CLASS_CATEGORY.......................................................... 3-19
  LOAD_KEY_MEMBER.................................................................... 3-20
  LOAD_ORG_ROLE........................................................................ 3-20
  LOAD_PROJECT........................................................................... 3-20
  Check Procedures..................................................................... 3-21
  CHECK_CHANGE_PROJECT_ORG_OK........................................... 3-21
  CHECK_DELETE_PROJECT_OK..................................................... 3-22
  CHECK_UNIQUE_PROJECT_REFERENCE........................................ 3-22

Project Definition Record and Table Datatypes......................... 3-22

Using Project Definition APIs..................................................... 3-54

Creating a Project Using the Load-Execute-Fetch APIs.................. 3-60

Structure APIs........................................................................... 3-66
  Structure API Views................................................................. 3-67
  Structure API Procedures......................................................... 3-68

Structure APIs Procedure Definitions........................................ 3-69
  ADD_TASK.................................................................................. 3-69
  APPLY_LP_PROG_ON_CWV......................................................... 3-72
  BASELINE_STRUCTURE............................................................... 3-72
  CHANGE_CURRENT_WORKING_VERSION..................................... 3-73
  CHANGE_STRUCTURE_STATUS..................................................... 3-73
  DELETE_STRUCTURE_VERSION.................................................... 3-74
EXEC_UPDATE_RBS .............................................................. 3-152
FREEZE_RBS_VERSION ....................................................... 3-153
ASSIGN_RBS_TO_PROJECT .................................................. 3-153

Dependency APIs ........................................................................ 3-155
Dependency API Views .............................................................. 3-155
Dependency API Procedures ..................................................... 3-155

Dependency API Procedure Definitions ................................. 3-155
CREATE_DEPENDENCY ......................................................... 3-155
UPDATE_DEPENDENCY ......................................................... 3-156
DELETE_DEPENDENCY ......................................................... 3-157

Task Assignment APIs ............................................................ 3-157
Task Assignment Views ............................................................. 3-157
Task Assignment API Procedures ............................................. 3-158
Task Assignment API Record and Table Datatype ....................... 3-158

Task Assignment API Procedure Definitions .......................... 3-162
LOAD_TASK_ASSIGNMENTS ............................................... 3-163
LOAD_TASK_ASSIGNMENT_PERIODS .................................... 3-163
EXECUTE_CREATE_TASK_ASGMTS .................................... 3-163
EXECUTE_UPDATE_TASK_ASGMTS .................................... 3-164
CREATE_TASK_ASSIGNMENTS ............................................. 3-164
CREATE_TASK_ASSIGNMENT_PERIODS .................................. 3-164
UPDATE_TASK_ASSIGNMENTS ............................................. 3-165
UPDATE_TASK_ASSIGNMENT_PERIODS .................................. 3-165
DELETE_TASK_ASSIGNMENTS ............................................. 3-166
FETCH_TASK_ASSIGNMENTS .............................................. 3-166
CONVERT_PM_TAREF_TO_ID ............................................... 3-167
INIT_TASK_ASSIGNMENTS ............................................... 3-167

4 Oracle Project Costing APIs

Asset APIs ................................................................................. 4-1
Asset API Views ................................................................. 4-1
Asset API Procedures and Functions ........................................ 4-2

Asset API Procedure and Function Definitions ........................ 4-2
ADD_PROJECT_ASSET ......................................................... 4-2
UPDATE_PROJECT_ASSET ...................................................... 4-3
DELETE_PROJECT_ASSET ...................................................... 4-4
ADD_ASSET_ASSIGNMENT ..................................................... 4-4
DELETE_ASSET_ASSIGNMENT ............................................... 4-4
LOAD_PROJECT_ASSET ......................................................... 4-5
LOAD_ASSET_ASSIGNMENT .................................................. 4-5
EXECUTE_ADD_PROJECT_ASSET.............................................................................. 4-6
CONVERT_PM_ASSETREF_TO_ID........................................................................ 4-7
FETCH_PROJECT_ASSET_ID................................................................................ 4-7
Cost Plus Application Programming Interface (API).......................................... 4-7
Get Burden Amount......................................................................................... 4-8
Example of Using the Cost Plus API.................................................................. 4-9

5 Oracle Project Billing APIs

Agreement and Funding APIs.............................................................................. 5-1
Security for Agreement and Funding APIs....................................................... 5-1
Agreement and Funding API Views.................................................................... 5-2
Agreement and Funding API Procedures.......................................................... 5-2
Agreement and Funding API Procedure Definitions........................................ 5-3
CREATE_AGREEMENT........................................................................................... 5-3
DELETE_AGREEMENT............................................................................................ 5-4
UPDATE_AGREEMENT........................................................................................... 5-5
CREATE_BASELINE_BUDGET................................................................................ 5-6
ADD_FUNDING...................................................................................................... 5-7
DELETE_FUNDING.................................................................................................. 5-8
UPDATE_FUNDING................................................................................................ 5-8
INIT_AGREEMENT.................................................................................................. 5-9
LOAD_AGREEMENT............................................................................................... 5-9
LOAD_FUNDING...................................................................................................... 5-10
EXECUTE_CREATE_AGREEMENT........................................................................... 5-10
EXECUTE_UPDATE_AGREEMENT.......................................................................... 5-11
FETCH_FUNDING.................................................................................................... 5-12
CLEAR_AGREEMENT.............................................................................................. 5-12
CHECK_DELETE_AGREEMENT_OK...................................................................... 5-12
CHECK_ADD_FUNDING_OK................................................................................... 5-13
CHECK_DELETE_FUNDING_OK.............................................................................. 5-14
CHECK_UPDATE_FUNDING_OK.............................................................................. 5-14
Using Agreement and Funding APIs.................................................................. 5-15
Creating an Agreement Using Load-Execute-Fetch APIs.................................... 5-19
Creating an Agreement Using a Composite Datatype API................................ 5-24
Event APIs........................................................................................................ 5-27
Event API Procedures....................................................................................... 5-27
Events API Record and Table Datatypes............................................................ 5-28
Event API Procedure Definitions..................................................................... 5-32
CREATE_EVENT................................................................................................... 5-32
DELETE_EVENT................................................................................................... 5-32
6 Oracle Project Management APIs

Project Deliverables APIs ................................................................. 6-1
  Project Deliverables API Procedures ............................................. 6-2

Budget APIs ......................................................................................... 6-7
  Budget API Views ........................................................................... 6-7
  Budget API Procedures .................................................................. 6-8

Budget Record and Table Datatypes .................................................... 6-9

Budget API Procedure Definitions ...................................................... 6-15
  ADD_BUDGET_LINE ........................................................................ 6-16
  BASELINE_BUDGET ..................................................................... 6-20
  CALCULATE_AMOUNTS ................................................................. 6-21
  CREATE_DRAFT_BUDGET .............................................................. 6-24
  CREATE_DRAFT_FINPLAN ................................................................. 6-27
  DELETE_BASELINE_BUDGET ......................................................... 6-30
  DELETE_BUDGET_LINE ................................................................. 6-33
  DELETE_DRAFT_BUDGET ............................................................... 6-37
  GET_PROJECT_ID ........................................................................... 6-39
  SET_PROJECT_ID ........................................................................... 6-39
  UPDATE_BUDGET ........................................................................... 6-39
  UPDATE_BUDGET_LINE ................................................................. 6-47
  CLEAR_BUDGET ............................................................................ 6-52
  CLEAR_CALCULATE_AMOUNTS ....................................................... 6-52
  EXECUTE_CALCULATE_AMOUNTS .................................................. 6-52
  EXECUTE_CREATE_DRAFT_BUDGET .............................................. 6-53
  EXECUTE_CREATE_DRAFT_FINPLAN ............................................. 6-54
  EXECUTE_UPDATE_BUDGET ........................................................... 6-54
  FETCH_BUDGET_LINE ................................................................. 6-55
  FETCH_CALCULATE_AMOUNTS ...................................................... 6-55
  INIT_BUDGET ............................................................................... 6-55
  INIT_CALCULATE_AMOUNTS .......................................................... 6-55
  LOAD_BUDGET_LINE ..................................................................... 6-55
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Budget APIs</td>
<td>6-56</td>
</tr>
<tr>
<td>Step 1: Connect to an Oracle Database</td>
<td>6-57</td>
</tr>
<tr>
<td>Step 2: Get the Budget Data</td>
<td>6-57</td>
</tr>
<tr>
<td>Step 3: Get Budget Line Data</td>
<td>6-59</td>
</tr>
<tr>
<td>Step 4: Interface Budget Information to the Server</td>
<td>6-62</td>
</tr>
<tr>
<td>Step 5: Start the Server-Side Process</td>
<td>6-63</td>
</tr>
<tr>
<td>Step 6: Retrieve Error Messages</td>
<td>6-63</td>
</tr>
<tr>
<td>Step 7: Finish the Load-Execute-Fetch Process</td>
<td>6-63</td>
</tr>
<tr>
<td>Creating a Budget Using the Load-Execute-Fetch APIs</td>
<td>6-63</td>
</tr>
<tr>
<td>Creating a Budget Using a Composite Datatype API</td>
<td>6-66</td>
</tr>
<tr>
<td>Refresh Planning Amounts API</td>
<td>6-69</td>
</tr>
<tr>
<td>Status APIs</td>
<td>6-70</td>
</tr>
<tr>
<td>Overview of Status API Views</td>
<td>6-71</td>
</tr>
<tr>
<td>Status API Views</td>
<td>6-74</td>
</tr>
<tr>
<td>Status API Procedures</td>
<td>6-76</td>
</tr>
<tr>
<td>Record and Table Datatypes</td>
<td>6-77</td>
</tr>
<tr>
<td>Required Parameters and Parameter Values</td>
<td>6-79</td>
</tr>
<tr>
<td>Status API Procedure Definitions</td>
<td>6-83</td>
</tr>
<tr>
<td>Custom Summarization Reporting APIs</td>
<td>6-87</td>
</tr>
<tr>
<td>Actuals Summarization API</td>
<td>6-87</td>
</tr>
<tr>
<td>Actuals Summarization API Procedures</td>
<td>6-88</td>
</tr>
<tr>
<td>Budget Summarization API</td>
<td>6-89</td>
</tr>
<tr>
<td>Project Performance Reporting APIs</td>
<td>6-90</td>
</tr>
</tbody>
</table>

**Part 3  ORACLE PROJECTS CLIENT EXTENSIONS**

7  **Overview of Client Extensions**

Client Extensions........................................................................................................... 7-1
Implementing Client Extensions.................................................................................... 7-2
  Analyzing Your Business Requirements.................................................................... 7-2
  Designing the Logic................................................................................................. 7-2
  Writing PL/SQL Procedures.................................................................................... 7-4

8  **Oracle Project Foundation Client Extensions**

Project Security Extension.......................................................................................... 8-1
Project Verification Extension...................................................................................... 8-3
Project and Task Date Extension............................................................................... 8-5
Project Workflow Extension......................................................................................... 8-8
Verify Organization Change Extension.................................................................................... 8-9
Transaction Import Extensions.......................................................................................... 8-10
  Pre-Import Client Extension for Internet Time.............................................................. 8-11
  Post-Import Client Extension for Internet Time............................................................ 8-13
Descriptive Flexfield Mapping Extension......................................................................... 8-14
Archive Project Validation Extension............................................................................... 8-18
Archive Custom Tables Extension.................................................................................. 8-18

9 Oracle Project Costing Client Extensions

Transaction Control Extensions........................................................................................ 9-1
  Case Study: New Charges Not Allowed......................................................................... 9-7
  Case Study: Organization-Based Transaction Controls.................................................. 9-9
  Case Study: Default Billable Status by Expenditure Type.............................................. 9-10

AutoApproval Extensions.............................................................................................. 9-11

Labor Costing Extensions.............................................................................................. 9-14

Labor Transaction Extensions.......................................................................................... 9-16

Overtime Calculation Extension...................................................................................... 9-24

Burden Costing Extension.............................................................................................. 9-28

Burden Resource Extension............................................................................................. 9-29

Allocation Extensions.................................................................................................. 9-30

Asset Allocation Basis Extension.................................................................................. 9-39

Asset Assignment Extension.......................................................................................... 9-41

Asset Lines Processing Extension.................................................................................. 9-42

Capital Event Processing Extension.............................................................................. 9-43

Capitalized Interest Extension....................................................................................... 9-44

CIP Grouping Extension................................................................................................. 9-46

CIP Account Override Extension................................................................................... 9-49

Depreciation Account Override Extension...................................................................... 9-50

Cross-Charge Client Extensions...................................................................................... 9-51
  Provider and Receiver Organizations Override Extension.......................................... 9-51
  Cross-Charge Processing Method Override Extension................................................ 9-52
  Transfer Price Determination Extension................................................................... 9-54
  Transfer Price Override Extension............................................................................. 9-55
  Transfer Price Currency Conversion Override Extension........................................ 9-57

10 Oracle Project Billing Client Extensions

Funding Revaluation Factor Extension........................................................................... 10-1

Billing Cycle Extension................................................................................................. 10-2

Billing Extensions.......................................................................................................... 10-3
Cost Accrual Billing Extension ................................................................. 10-33
Cost Accrual Identification Extension ....................................................... 10-34
Labor Billing Extensions ........................................................................... 10-35
Non-Labor Billing Extensions ................................................................. 10-38
Retention Billing Extension ..................................................................... 10-40
Automatic Invoice Approve/Release Extension ....................................... 10-40
Output Tax Extension ............................................................................... 10-42
Receivables Installation Override Extension ......................................... 10-43
AR Transaction Type Extension .............................................................. 10-45

11 Oracle Project Resource Management Client Extensions
Assignment Approval Changes Extension .................................................. 11-1
Assignment Approval Notification Extension ......................................... 11-2
Candidate Notification Workflow Extension .......................................... 11-4

12 Oracle Project Management Client Extensions
Workplan Workflow Extension ................................................................. 12-1
Progress Management Extension ............................................................ 12-2
Budget Calculation Extensions ............................................................... 12-3
Budget Verification Extension ................................................................. 12-7
Budget Workflow Extension .................................................................. 12-8
Estimate to Complete Generation Method Extension ............................... 12-9
Control Item Document Numbering Extension ....................................... 12-10
Issue and Change Workflow Extension .................................................. 12-11
Project Status Report Workflow Extension ............................................ 12-12
Custom Performance Measure Extension .............................................. 12-13
Project Performance Status Extension .................................................... 12-14
Project Status Inquiry (PSI) Extension .................................................... 12-15
Project Status Inquiry Burdening Commitments Extension ...................... 12-18
Project Status Inquiry Commitment Changes Extension ........................ 12-19

Part 4 ORACLE PROJECTS OPEN INTERFACES

13 Oracle Projects Open Interfaces
Transaction Import .................................................................................. 13-1
Transaction Import Process Diagram ....................................................... 13-2
Using Transaction Import ...................................................................... 13-4
Importing Transactions .......................................................................... 13-6
Types of Items That You Can Import ...................................................... 13-7
Loading Items as Costed or Uncosted .......................................................... 13-8
Loading Items as Accounted or Unaccounted.............................................. 13-9
Loading Burden Transactions..................................................................... 13-10
Loading Project Manufacturing Costs.......................................................... 13-10
Loading Foreign Currency Transactions...................................................... 13-12
Import Options (Transaction Source).......................................................... 13-15
Grouping Transactions into Expenditure Batches and Expenditures.............. 13-16
Transaction Import Example: Labor and Expense by Employee Number........ 13-19
Transaction Import Example: Usage.............................................................. 13-21
Viewing and Processing Imported Transactions.......................................... 13-23

Transaction Import Interface........................................................................ 13-25
Transaction Import Validation...................................................................... 13-25
Target Expenditure Tables........................................................................... 13-26
The Transaction Import Interface Table........................................................ 13-27
PA_TRANSACTION_INTERFACE_ALL Column Requirements.......................... 13-28
Resolving Import Exceptions........................................................................ 13-53

Index
Send Us Your Comments

Oracle Projects APIs, Client Extensions, and Open Interfaces Reference, Release 12
Part No. B25624-02

Oracle welcomes customers’ comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on Oracle MetaLink and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.
Preface

Intended Audience

Welcome to Release 12 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.

See Related Information Sources on page xix for more Oracle Applications product information.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

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. Accessibility 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 more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.
Accessibility of Code Examples in Documentation

Screen readers 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, some screen readers 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.

Structure

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.

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.

3 Oracle Project Foundation APIs
This chapter describes how to implement APIs for:

- Project and task information
- Structure information
- Resource list and resource list member information
- Dependency information
- Task Assignment information

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

5 Oracle Project Billing APIs
This chapter describes how to implement APIs for:

- Agreements and funding
- Events

6 Oracle Project Management APIs
This chapter describes how to implement Oracle Project Management APIs.
7 Overview of Client Extensions
This chapter describes everything you need to know about designing and writing client extensions in Oracle Projects.

8 Oracle Project Foundation Client Extensions
This chapter describes the client extensions in the Oracle Project Foundation application.

9 Oracle Project Costing Client Extensions
This chapter describes the client extensions in the Oracle Project Costing application.

10 Oracle Project Billing Client Extensions
This chapter describes the client extensions in the Oracle Project Billing application.

11 Oracle Project Resource Management Client Extensions
This chapter describes the client extensions in the Oracle Project Resource Management application.

12 Oracle Project Management Client Extensions
This chapter describes the client extensions in the Oracle Project Management application.

13 Oracle Projects Open Interfaces
This chapter describes the open interfaces in the Oracle Projects applications.

Related 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.

Integration Repository
The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite’s business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

Online Documentation
Oracle Applications documentation, including online help patches (HTML) and guides (PDF), is 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 programs.

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 Portfolio Analysis). 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 Billing User Guide

This guide shows you how to use Oracle Project Billing to define revenue and invoicing rules for your projects, generate revenue, create invoices, and integrate with other Oracle Applications to process revenue and invoices, process client invoicing, and measure the profitability of your contract projects.

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 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 Portfolio Analysis User Guide
This guide contains the information you need to understand and use Oracle Project Portfolio Analysis. It includes information about project portfolios, planning cycles, and metrics for ranking and selecting projects for a project portfolio.

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.

Oracle Projects Glossary
This glossary provides definitions of terms that are shared by all Oracle Projects applications. If you are unsure of the meaning of a term you see in an Oracle Projects guide, please refer to the glossary for clarification. You can find the glossary in the online help for Oracle Projects, and in the Oracle Projects Fundamentals book.

User Guides Related to This Product

Oracle Assets User Guide
This guide provides you with information on how to implement and use Oracle Assets. Use this guide to understand the implementation steps required for application use, including defining depreciation books, depreciation method, and asset categories. It also contains information on setting up assets in the system, maintaining assets, retiring and reinstating assets, depreciation, group depreciation, accounting and tax accounting, budgeting, online inquiries, impairment processing, and Oracle Assets reporting. This guide also includes a comprehensive list of profile options that you can set to customize application behavior.

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

Oracle Financials Implementation Guide
This guide provides you with information on how to implement the Oracle Financials E-Business Suite. It guides you through setting up your organizations, including legal entities, and their accounting, using the Accounting Setup Manager. It covers intercompany accounting and sequencing of accounting entries, and it provides examples.
**Oracle General Ledger Implementation Guide**
This guide provides information on how to implement Oracle General Ledger. Use this guide to understand the implementation steps required for application use, including how to set up Accounting Flexfields, Accounts, and Calendars.

**Oracle General Ledger User's Guide**
This guide provides you with information on how to use Oracle General Ledger. Use this guide to learn how to create and maintain ledgers, ledger currencies, budgets, and journal entries. This guide also includes information about running financial reports.

**Oracle Grants Accounting User Guide**
This guide provides you with information about how to implement and use Oracle Grants Accounting. Use this guide to understand the implementation steps required for application use, including defining award types, award templates, allowed cost schedules, and burden set up. This guide also explains how to use Oracle Grants Accounting to track grants and funded projects from inception to final reporting.

**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:

- **Oracle HRMS Enterprise and Workforce Management Guide**
  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 Internet Expenses Implementation and Administration Guide**
This book explains in detail how to configure Oracle Internet Expenses and describes its integration with other applications in the E-Business Suite, such as Oracle Payables and Oracle Projects. Use this guide to understand the implementation steps required for application use, including how to set up policy and rate schedules, credit card policies, audit automation, and the expenses spreadsheet. This guide also includes detailed information about the client extensions that you can use to extend Oracle Internet Expenses functionality.
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 Implementation Guide
This guide provides you with information on how to implement Oracle Payables. Use this guide to understand the implementation steps required for how to set up suppliers, payments, accounting, and tax.

Oracle Payables User’s Guide
This guide describes how to use Oracle Payables to create invoices and make payments. In addition, it describes how to enter and manage suppliers, import invoices using the Payables open interface, manage purchase order and receipt matching, apply holds to invoices, and validate invoices. It contains information on managing expense reporting, procurement cards, and credit cards. This guide also explains the accounting for Payables transactions.

Oracle Payments Implementation Guide
This guide describes how Oracle Payments, as the central payment engine for the Oracle E-Business Suite, processes transactions, such as invoice payments from Oracle Payables, bank account transfers from Oracle Cash Management, and settlements against credit cards and bank accounts from Oracle Receivables. This guide also describes how Oracle Payments is integrated with financial institutions and payment systems for receipt and payment processing, known as funds capture and funds disbursement, respectively. Additionally, the guide explains to the implementer how to plan the implementation of Oracle Payments, how to configure it, set it up, test transactions, and how to use it with external payment systems.

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

Oracle Property Manager Implementation Guide
Use this guide to learn how to implement Oracle Property Manager and perform basic setup steps such as setting system options and creating lookup codes, contacts, milestones, grouping rules, term templates, and a location hierarchy. This guide also describes the setup steps that you must complete in other Oracle applications before you can use Oracle Property Manager.
Oracle Property Manager User Guide

Use this guide to learn how to use Oracle Property Manager to create and administer properties, space assignments, and lease agreements.

Oracle Public Sector Advanced Features User Guide

Oracle Public Sector Advanced Features is an overlay of features that extend the existing functionality of Oracle Financials for the specific needs of the Public Sector. This guide provides information about setting up and using Oracle Public Sector Advanced Features. These features include multi-fund accounts receivable, encumbrance reconciliation reports, Governmental Accounting Standards Board (GASB) 34/35 asset accounting, enhanced funds available inquiry, the Funds Available Detail report, and the Funds Check API.

Oracle Purchasing User's Guide

This guide describes how to create and approve purchasing documents, including requisitions, different types of purchase orders, quotations, RFQs, and receipts. This guide also describes how to manage your supply base through agreements, sourcing rules, and approved supplier lists. In addition, this guide explains how you can automatically create purchasing documents based on business rules through integration with Oracle Workflow technology, which automates many of the key procurement processes.

Oracle Receivables User Guide

This guide provides you with information on how to use Oracle Receivables. Use this guide to learn how to create and maintain transactions and bills receivable, enter and apply receipts, enter customer information, and manage revenue. This guide also includes information about accounting in Receivables. Use the Standard Navigation Paths appendix to find out how to access each Receivables window.

Oracle Subledger Accounting Implementation Guide

This guide provides setup information for Oracle Subledger Accounting features, including the Accounting Methods Builder. You can use the Accounting Methods Builder to create and modify the setup for subledger journal lines and application accounting definitions for Oracle subledger applications. This guide also discusses the reports available in Oracle Subledger Accounting and describes how to inquire on subledger journal entries.

Oracle Time & Labor Implementation and User Guide

This guide describes how to capture work patterns such as shift hours so that this information can be used by other applications such as Oracle General Ledger and Oracle Projects.
BIS 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.

Installation and System Administration

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications. It is 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. 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.

Oracle Applications Upgrade Guide: Release 11i to Release 12

Refer to this guide if you are upgrading your Oracle Applications Release 11i products to Release 12. This guide describes the upgrade process and lists database and product-specific upgrade tasks. You must be at Release 11i to upgrade to Release 12. You cannot upgrade to Release 12 directly from releases prior to 11i.

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 programs.
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

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.

**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, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

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

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form 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.
Part 1

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

This chapter covers the following topics:

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

## 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 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 write. 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.
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.

This chapter covers the following topics:

- Introduction to Oracle Projects APIs
- Overview of the Oracle Projects APIs
- Integrating Your External System with Oracle Projects
- Security Requirements
- Handling Error Messages
- Standard API Parameters
- Common APIs
- Controlling Actions in Oracle Projects
- Using API Procedures

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 product is no longer licensed. 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 2-30.)

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

<table>
<thead>
<tr>
<th>Information That Originates in Oracle Projects</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project templates with Quick Entry (overridable) fields</td>
<td>You can override some of the template’s default values when you create a project.</td>
</tr>
</tbody>
</table>

Resources

Organizations

Calendars (both GL and PA periods)

Estimate to Complete (planned for a future release)
<table>
<thead>
<tr>
<th>Information That Originates in Oracle Projects</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actuals: cost amounts (raw and burdened), commitments (raw and burdened), quantities, revenue, PA or GL period, inception-to-date, period-to-date</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Information That Originates in Your Project Management System</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects and tasks of the work breakdown structure (WBS)</td>
<td>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.</td>
</tr>
<tr>
<td>Budgets: Types, Time-Phased, Amounts, Quantities, Baseline</td>
<td></td>
</tr>
<tr>
<td>Schedules and schedule changes</td>
<td></td>
</tr>
<tr>
<td>Task parent reassignment</td>
<td>You can reassign a task to a different parent task as long the reassigned task remains under the same top task.</td>
</tr>
<tr>
<td>Percent complete: project level, WBS (any level)</td>
<td>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.</td>
</tr>
</tbody>
</table>
Information That Originates in Your Project Management System

<table>
<thead>
<tr>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

Earned value progress reporting: Budgeted Cost of Work Scheduled, Budgeted Cost of Work Performed, Actual Cost of Work Performed, Budget at Completion

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 \texttt{paamgcrrole.sql} to create and assign these database roles. The script resides in the \texttt{$PA\_TOP/sql} directory on the server and creates an output file called \texttt{paamgcrrole.lst}. Run the script from any directory in which you have write privileges. You run this script as any user, such as SYSTEM or SYS with Create Role and Grant privileges on Oracle Projects Public APIs and Views.

The script requires the following arguments:

- New database role name, such as PMXFACE
- Username for the Oracle Applications user account, such as APPS

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

```
start $PA_TOP/sql/paamgcrrole.sql &role &un_apps
```

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

```
start $PA_TOP/sql/paamgcrrole.sql PMXFACE APPS
```

The script creates the role, and grants the necessary privileges on the required database objects. Check the file \texttt{paamgcrrole.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 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 `paamgcuser.sql` to create database users. The script resides in the `$PA_TOP/sql` directory on the server. The script creates an output file called `paamgcuser.lst`. Run the script from any directory in which you have write privileges. You run this script as any user, such as SYSTEM or SYS with a Create User and Create Synonym privileges on Oracle Projects Public APIs.

The script requires the following arguments:

- Existing database role name, such as PMXFACE. You must use the same role name that you created in Step 1.

- Username for the Oracle Applications user account, such as APPS

- Proposed new database username

- Proposed new database user password

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

```
start $PA_TOP/sql/paamgcuser.sql &role &un_apps &uname &pwd
```

For example, to create the user JCLARK with a password of WELCOME, enter:

```
start $PA_TOP/sql/paamgcuser.sql PMXFACE APPS JCLARK WELCOME
```

Check the file `paamgcuser.lst` to ensure that the script completed successfully.

Template Script to Create Database Users

Oracle Projects provides a template script, `$PA_TOP/sql/paamgustemp.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. You run this script as any user, such as SYSTEM or SYS, with a Create User and Create Synonym privileges on Oracle Projects Public 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.

Restoring Grants to the Database

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 `paamgurole.sql` (located in `$PA_TOP/sql`) to restore the grants. You run this script as any user, such as SYSTEM or SYS, that has Grant privileges on Oracle Projects Public APIs and Views.

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

```
start $PA_TOP/sql/paamgurole.sql &role &un_apps
```

For example:

```
start $PA_TOP/sql/paamgurole.sql PMXFACE APPS
```

Check the file `paamgurole.lst` to ensure that the script completed successfully.

Migrating Existing Roles from Previous Releases to Release 12

To migrate existing database roles and database users to use Oracle Projects Public APIs, run the scripts `paamgurole.sql` and `paamguuser.sql` in order. These scripts are located in the `$PA_TOP/sql` directory. You run these scripts as any user, such as SYSTEM or SYS, that has Grant privileges and Create Synonyms privileges on Oracle Projects Public APIs and Views.

From a SQL*Plus session, use the following syntax to run the `paamgurole.sql` script:

```
start $PA_TOP/sql/paamgurole.sql &role &un_apps
```

For example:

```
start $PA_TOP/sql/paamgurole.sql PMXFACE APPS
```

Check the file `paamgurole.lst` to ensure that the script completed successfully.

From a SQL*Plus session, use the following syntax to run the `paamguuser.sql` script:

```
start $PA_TOP/sql/paamguuser.sql &un_apps &name &role
```

For example:
Check the file `paamguuser.lst` to ensure that the script completed successfully.

**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`.

*Note:* You can see an updated version of the view definitions in Oracle's electronic Technical Reference Manual web page.

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:

```sql
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Usage</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_API_VERSION_NUMBER</td>
<td>IN</td>
<td>NUMBER</td>
<td>Yes</td>
<td>API standard</td>
</tr>
<tr>
<td>P_RESPONSIBILITY_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The reference code that uniquely identifies the chosen responsibility (refer to Step 2: Choose a Responsibility, page 2-8)</td>
</tr>
<tr>
<td>P_USER_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The identification code of the corresponding user returned by the view (refer to Step 2: Choose a Responsibility, page 2-8)</td>
</tr>
<tr>
<td>P_MSG_COUNT</td>
<td>OUT NOCOPY</td>
<td>NUMBER</td>
<td></td>
<td>API standard</td>
</tr>
<tr>
<td>P_MSG_DATA</td>
<td>OUT NOCOPY</td>
<td>VARCHAR2</td>
<td>(2000)</td>
<td>API standard</td>
</tr>
<tr>
<td>P_RETURN_STATUS</td>
<td>OUT NOCOPY</td>
<td>VARCHAR2</td>
<td>(1)</td>
<td>Return status. Valid values are: S (Success), E (Error), and U (Unexpected error).</td>
</tr>
<tr>
<td>Parameter</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>-----------------</td>
<td>------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_RESP_APPL_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the responsibility application</td>
</tr>
<tr>
<td>P_ADVANCED_PROJ_SEC_FLAG</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates whether to use role-based security (Default = N)</td>
</tr>
<tr>
<td>P_CALLING_MODE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Calling mode</td>
</tr>
<tr>
<td>P_OPERATING_UNIT_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Sets the ORG context</td>
</tr>
</tbody>
</table>

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

**Note:** If you are not implementing Multiple Organization Access Control, you do not need to set the P_OPERATING_UNIT_ID parameter. In that case, the P_OPERATING_UNIT_ID parameter accepts the default MO: Operating Unit value assigned to the responsibility.

### 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 2-25.

API Messages

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

<table>
<thead>
<tr>
<th>New Message Code</th>
<th>Length</th>
<th>Description</th>
<th>Token(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_API_CONV_ERROR</td>
<td>21</td>
<td>You entered an invalid API parameter.</td>
<td>ATTR_NAME, ATTR_VALUE</td>
</tr>
<tr>
<td>_AMG</td>
<td></td>
<td>Please enter a valid parameter and try again.</td>
<td></td>
</tr>
<tr>
<td>PA_ALL_WARN_NO</td>
<td>27</td>
<td>This user is not yet registered as an employee.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE,</td>
</tr>
<tr>
<td>_EMPL_REC_AMG</td>
<td></td>
<td></td>
<td>RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_BU_AMT_ALLOC_LT</td>
<td>27</td>
<td>Total amount allocated cannot be less than amount accrued or billed.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, START_DATE, RESOURCE_NAME,</td>
</tr>
<tr>
<td>_ACCR_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_BU_BASE_RES_LIST</td>
<td>30</td>
<td>You cannot change the resource list for a baselined budget</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME,</td>
</tr>
<tr>
<td>_EXISTS_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_BU_CORE_NO_VERSION_ID_AMG</td>
<td>28</td>
<td>A budget does not exist for this project with specified budget type.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_BU_INVALID_NEW_PERIOD_AMG</td>
<td>28</td>
<td>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).</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_BU_NO_BUDGET_AMG</td>
<td>20</td>
<td>There are no budget lines in this draft budget. The budget must be entered before baseline.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_BU_NO_PROJ_END_DATE_AMG</td>
<td>26</td>
<td>Project does not have a start date or a completion date.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_BU_NO_TASK_PROJ_DATE_AMG</td>
<td>27</td>
<td>Task does not have a start date or a completion date.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_BU_UNBAL_PROJ_BUDG_AMG</td>
<td>25</td>
<td>Project funding is not equal to the budget total. To baseline a draft budget, the budget total must be as same as funding total.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_BU_UNBAL_TASK_BUDG_AMG</td>
<td>25</td>
<td>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.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>PA_COPY_PROJECT_FAILED_AMG</td>
<td>26</td>
<td>Error occurred while creating the project.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>PA_CREATE_CONTACTS_FAILED_AMG</td>
<td>29</td>
<td>Error occurred while creating Customer Contact information.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_CUST_NOT_OVERRIDABLE_AMG</td>
<td>27</td>
<td>You cannot override the Customer field while using this template.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_DESC_NOT_OVERRIDABLE_AMG</td>
<td>27</td>
<td>You cannot override the Description field while using this template.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_GET_CUST_INFO_FAILED_AMG</td>
<td>27</td>
<td>Error occurred while getting Customer information.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_HAS_REV_INV_AMG</td>
<td>18</td>
<td>Distribution rule cannot be changed because cost/revenue/invoices exist. Cause: You cannot change the distribution rule because the project has costed items, revenue, or invoices.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_INVALID_DIST_RULE_AMG</td>
<td>24</td>
<td>Distribution Rule is invalid.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_INVALID_ORG_AMG</td>
<td>18</td>
<td>Organization is invalid.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_INVALID_PT_CLASS_ORG_AMG</td>
<td>27</td>
<td>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.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_NO_BILL_TO_ADDRESS_AMG</td>
<td>25</td>
<td>Active primary Bill To Address does not exist for the specified customer.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_NO_BILL_TO_CONTACT_AMG</td>
<td>25</td>
<td>Active primary billing contact does not exist for the specified customer.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_NO_CLIENT_EXISTS_AMG</td>
<td>23</td>
<td>The billing allocation across project client(s) is incomplete.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_NO_CONTACT</td>
<td>28</td>
<td>Billing contact not defined for each customer.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXISTS_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_MANAGER</td>
<td>24</td>
<td>Project manager not currently defined for this project.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_ORIG_PROJ_ID</td>
<td>22</td>
<td>Original project ID is not specified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_PROJ_CREATED</td>
<td>22</td>
<td>New project not created. No project information in the source project.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_PROJ_ID_AMG</td>
<td>17</td>
<td>Project ID not specified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_NO_REQ_CATEGORY</td>
<td>29</td>
<td>All mandatory class categories have not been classified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXISTS_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_SHIP_TO</td>
<td>25</td>
<td>Active primary Ship To Address does not exist for the specified customer.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_ADDRESS_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_TASK_COPIED</td>
<td>21</td>
<td>No task is copied because there are tasks in the source project.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_TASK_ID_D_AMG</td>
<td>19</td>
<td>You cannot delete this task because no task information has been provided.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>PA_NO_TASK_ID_ST_AMG</td>
<td>20</td>
<td>You cannot create a subtask below this task because task information was not specified.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>PA_NO_TOP_TASK_ID</td>
<td>25</td>
<td>You cannot create a subtask below this task because task does not have top task ID.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_NO_UNIQUE_ID_AMG</td>
<td>19</td>
<td>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.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>PA_PRODUCT_CODE</td>
<td>30</td>
<td>External product code required.</td>
<td>General</td>
</tr>
<tr>
<td>_IS_MISSING_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJECT_NAME</td>
<td>30</td>
<td>Project name required.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_IS_MISSING_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJECT_REF</td>
<td>29</td>
<td>External project reference required.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_IS_MISSING_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJECT_STATUS</td>
<td>29</td>
<td>The project status is invalid.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_INVALID_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_AP_INV</td>
<td>26</td>
<td>You cannot delete this project because supplier invoices exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_BUDGET</td>
<td>26</td>
<td>You cannot delete this project because budgets exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_BURDEN</td>
<td>29</td>
<td>The project is being used for the purpose of accumulating burden costs on</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_SUM_DEST_D_AMG</td>
<td></td>
<td>project types.</td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_CMT_TXN</td>
<td>27</td>
<td>You cannot delete this project because project commitment transactions exist.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_EVENT_EXIST</td>
<td>25</td>
<td>You cannot delete this project because events exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_EXP_ITEM</td>
<td>28</td>
<td>You cannot delete this project because expenditure items exist.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_FUND_EXIST</td>
<td>25</td>
<td>You cannot delete this project because funding exists.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_INV_DIST</td>
<td>28</td>
<td>You cannot delete this project because supplier invoice distribution lines</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td>exist</td>
<td></td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>PA_PROJ_IN_USE</td>
<td>29</td>
<td>You cannot delete this project because project references exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXTERNAL_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_ORG_NOT</td>
<td>26</td>
<td>This project organization is not active or is not within the current Project/Task owning organization hierarchy.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_ACTIVE_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PROJ_PO_DIST</td>
<td>27</td>
<td>You cannot delete this project because purchase order distributions exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_COM_RUL_SET</td>
<td>29</td>
<td>You cannot delete this project because compensation rules exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_CREATED_REF</td>
<td>29</td>
<td>You cannot delete this project because compensation rule sets exist</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_INSUF_BILL</td>
<td>28</td>
<td>Billing contact not defined for each customer.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_CONTACT_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_INSUF_CLASS</td>
<td>27</td>
<td>You must specify all mandatory class categories.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_CODES_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_INSUF_PROJ</td>
<td>24</td>
<td>Project manager not currently defined for this project.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_MGR_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_INVALID_START</td>
<td>28</td>
<td>Project start date must be earlier than all task start dates.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_DATE_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_NAME_NOT</td>
<td>27</td>
<td>Project name must be unique across all operating units in the Oracle Applications installation.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_UNIQUE_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PR_NAME_NOT</td>
<td>25</td>
<td>Project name must be unique across all operating units in the Oracle Applications installation.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>_UNIQUE_A_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>PA_PR_NO_PROJ_NAME_AMG</td>
<td>22</td>
<td>Project name not specified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_NO_PROJ_NUM_AMG</td>
<td>21</td>
<td>Project number ID not specified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_NO_UPD_SEGMENT1_EXP_AMG</td>
<td>29</td>
<td>You cannot change the project number because expenditure items exist.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_NUMERIC_NUM_REG_AMG</td>
<td>25</td>
<td>Please enter a numeric project number.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_NUMERIC_NUM_REQ_AMG</td>
<td>25</td>
<td>Your implementation requires a numeric project number.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_NUM_NOT_UNIQUE_AMG</td>
<td>26</td>
<td>Project number must be unique across all operating units in the Oracle Applications installation.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_NUM_NOT_UNIQUE_A_AMG</td>
<td>24</td>
<td>Project number must be unique across all operating units in the Oracle Applications installation.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_PO_REQ_DIST_EXIST_D_AMG</td>
<td>29</td>
<td>You cannot delete this project because purchase order requisitions exist.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_START_DATE_NEEDED_AMG</td>
<td>27</td>
<td>The start date of the project is required if the completion date of the project is specified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PR_START_DATE_NEEDED_AMG</td>
<td>23</td>
<td>The start date of the project is required if the completion date of the project is specified.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_PUBLIC_SECTOR_INVALID_AMG</td>
<td>28</td>
<td>Invalid value for Public Sector flag.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------</td>
<td>-------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>PA_RE_ASSGMENT_NOT_FOUND_AMG</td>
<td>26</td>
<td>Resource list assignment not found.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_RE_PROJ_NOT_FOUND_AMG</td>
<td>24</td>
<td>Specified project is invalid.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_RE_RL_INACTIVE_AMG</td>
<td>21</td>
<td>Resource list is not active.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_RE_RL_NOT_FOUND_AMG</td>
<td>22</td>
<td>Specified resource list is invalid.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_RE_USE_CODE_NOT_FOUND_AMG</td>
<td>28</td>
<td>Use code not found.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_SOURCE_TEMPLATE_INVALID_AMG</td>
<td>30</td>
<td>Source template ID is invalid.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_SOURCE_TEMP_IS_MISSING_AMG</td>
<td>30</td>
<td>Source template ID is required.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_SU_INVALID_DATES_AMG</td>
<td>23</td>
<td>From Date must be on or before the To Date.</td>
<td>PROJECT_NUMBER</td>
</tr>
<tr>
<td>PA_TASK_BURDEN_SUM_DEST_ST_AMG</td>
<td>30</td>
<td>The task is being used for the purpose of accumulating burden costs on project types.</td>
<td>PROJECT_NUMBER, TASK_NUMBER</td>
</tr>
<tr>
<td>PA_TASK_BURDEN_SUM_DEST_ST_AMG</td>
<td>29</td>
<td>The task is being used for the purpose of accumulating burden costs on project types.</td>
<td>PROJECT_NUMBER, TASK_NUMBER</td>
</tr>
<tr>
<td>PA_TASK_FUND_NO_PROJ_EVT_AMG</td>
<td>28</td>
<td>Task funding with project level events is not allowed.</td>
<td>PROJECT_NUMBER, TASK_NUMBER, BUDGET_TYPE, RESOURCE_NAME, START_DATE</td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>PA_TASK_IN_USE</td>
<td>26</td>
<td>You cannot delete this task because task references exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXTERNAL_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_AP_INV_DIST</td>
<td>30</td>
<td>You cannot delete this task because invoice distribution lines exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_AP_INV_DIST</td>
<td>30</td>
<td>You cannot create a subtask below this task because supplier invoice distribution lines exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_AP_INV_EXIST</td>
<td>25</td>
<td>You cannot delete this task because supplier invoices exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_AP_INV_EXIST</td>
<td>26</td>
<td>You cannot create a subtask below this task because supplier invoices exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_ASSETASSIG</td>
<td>30</td>
<td>You cannot create a subtask below this task because assets have been assigned.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_BUDGET</td>
<td>25</td>
<td>You cannot delete this task because budgets exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_BUDGET</td>
<td>26</td>
<td>You cannot create a subtask below this task because budgets exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_BUR_SCHOVR</td>
<td>30</td>
<td>You cannot create a subtask below this task because burden schedule overrides exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_CMT_TXN</td>
<td>26</td>
<td>You cannot delete this task because commitment transactions exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_EBILL_RATE</td>
<td>30</td>
<td>You cannot create a subtask below this task because employee billing rate overrides exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Message Code</td>
<td>Length</td>
<td>Description</td>
<td>Token(s)</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>PA_TSK_EVENT_EXIST</td>
<td>24</td>
<td>You cannot delete this task because events exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_EXP_ITEM_EXIST</td>
<td>27</td>
<td>You cannot delete this task because expenditure items exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_EXP_ITEM_EXIST</td>
<td>28</td>
<td>You cannot create a subtask below this task because expenditure items exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_FUND_EXIST</td>
<td>27</td>
<td>You cannot delete this task because supplier invoice distribution line exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_JBILLTITLE</td>
<td>30</td>
<td>You cannot create a subtask below this task because job billing title overrides exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_JBILL_RATE</td>
<td>30</td>
<td>You cannot create a subtask below this task because job bill rate overrides exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_LAB_MULT</td>
<td>26</td>
<td>You cannot create a subtask below this task because there is labor multiplier for this task.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_L_COST_MUL</td>
<td>30</td>
<td>You cannot create a subtask below this task because labor cost multipliers exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_NL_BIL_RAT</td>
<td>30</td>
<td>You cannot create a subtask below this task because non-labor bill rate overrides exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_PO_DIST</td>
<td>26</td>
<td>You cannot delete this task because supplier invoice distribution lines exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_D_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_TSK_PO_DIST</td>
<td>27</td>
<td>You cannot create a subtask below this task because supplier invoice distribution lines exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>_EXIST_ST_AMG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## New Message Code Length Description Token(s)

<table>
<thead>
<tr>
<th>New Message Code</th>
<th>Length</th>
<th>Description</th>
<th>Token(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TSK_PO_REQDIST_EXIST_ST_AMG</td>
<td>30</td>
<td>You cannot create a subtask below this task because purchase order requisitions exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>PA_TSK_RULE_SET_EXIST_D_AMG</td>
<td>27</td>
<td>You cannot delete this task because compensation rule sets exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
<tr>
<td>PA_TSK_TXN_CONT_EXIST_ST_AMG</td>
<td>28</td>
<td>You cannot create a subtask below this task because transaction controls exist.</td>
<td>PROJECT_NUMBER, TASK__NUMBER</td>
</tr>
</tbody>
</table>

## 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Usage</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_COMMIT</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>T = The API issues the commit to the database. Default = F (False)</td>
</tr>
<tr>
<td>P_INIT_MSG_LIS</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>Set this parameter to T (True) if you want to initialize the global message table. Default = F (False)</td>
</tr>
<tr>
<td>P_API_VERSION</td>
<td>IN</td>
<td>NUMBER</td>
<td>Yes</td>
<td>For the current version of the APIs, this parameter must be set to 1.0. This may change in future versions of the APIs.</td>
</tr>
<tr>
<td>P_RETURN</td>
<td>OUT NOCOPY</td>
<td>VARCHAR2</td>
<td></td>
<td>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)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>----------</td>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_MSG_COUNT</td>
<td>OUT NOCOPY</td>
<td>NUMBER</td>
<td></td>
<td>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 &gt; 1, you must use the GET_MESSAGES API to retrieve the messages.</td>
</tr>
<tr>
<td>P_MSG_DATA</td>
<td>OUT NOCOPY</td>
<td>VARCHAR2 (2000)</td>
<td></td>
<td>Holds the message code, if the API returned only one error/warning message. Otherwise, the column is left blank.</td>
</tr>
</tbody>
</table>

**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_pm_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_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.
```sql
--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.

**Important:** 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.

```sql
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.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Usage</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_ENCODED</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>T: Return message code; F (Default): Return message text</td>
</tr>
<tr>
<td>P_MSG_COUNT</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The message 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.</td>
</tr>
<tr>
<td>P_MSG_DATA</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>The P_MSG_DATA value returned by the API that raised the error</td>
</tr>
<tr>
<td>P_DATA</td>
<td>OUT</td>
<td>VARCHAR2</td>
<td></td>
<td>The message code (if P_ENCODED = T) or the message text (if P_ENCODED = F)</td>
</tr>
<tr>
<td>P_MSG_INDEX_OUT</td>
<td>OUT</td>
<td>NUMBER</td>
<td></td>
<td>The index (cell) of the message in the global message stack</td>
</tr>
<tr>
<td>P_MSG_INDEX</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Message index number (default = 1)</td>
</tr>
</tbody>
</table>

**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.initialize 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:

```sql
-- Initialize the message stack
FND_MSG_PUB.initialize;
p_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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_DEF_CHAR</td>
<td>OUT NOCOPY</td>
<td>VARCHAR2(3)</td>
<td>Returns the default value for character variables</td>
</tr>
</tbody>
</table>
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:

```plsql
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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Usage</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_API_VERSION_NUMBER</td>
<td>IN</td>
<td>NUMBER</td>
<td>Y</td>
<td>API standard</td>
</tr>
<tr>
<td>p_MSG_COUNT</td>
<td>OUT</td>
<td>NUMBER</td>
<td>NOCOPY</td>
<td>API standard</td>
</tr>
<tr>
<td>Parameter</td>
<td>Usage</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>p_MSG_DATA</td>
<td>OUT</td>
<td>VARCHAR2</td>
<td>NOCOPY</td>
<td>API standard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_RETURN_STATUS</td>
<td>OUT</td>
<td>VARCHAR2</td>
<td>NOCOPY</td>
<td>API standard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_PROJECT_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>Y</td>
<td>Unique identifier of the project</td>
</tr>
<tr>
<td>p_LAST_ACCUM_PERIOD</td>
<td>OUT</td>
<td>VARCHAR2</td>
<td>NOCOPY</td>
<td>The period up to which the project has been summarized</td>
</tr>
<tr>
<td>p_LAST_ACCUM_START_DATE</td>
<td>OUT</td>
<td>DATE</td>
<td>NOCOPY</td>
<td>The start date of the last summarized period</td>
</tr>
<tr>
<td>p_LAST_ACCUM_END_DATE</td>
<td>OUT</td>
<td>DATE</td>
<td>NOCOPY</td>
<td>The end date of the last summarized period</td>
</tr>
<tr>
<td>p_CURRENT_REPORTING</td>
<td>OUT</td>
<td>VARCHAR2</td>
<td>NOCOPY</td>
<td>The PA period that is defined in the current reporting period</td>
</tr>
<tr>
<td>_PERIOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_CURRENT_PERIOD_START</td>
<td>OUT</td>
<td>DATE</td>
<td>NOCOPY</td>
<td>The start date of the current reporting period</td>
</tr>
<tr>
<td>_DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p_CURRENT_PERIOD_END</td>
<td>OUT</td>
<td>DATE</td>
<td>NOCOPY</td>
<td>The end date of the current reporting period</td>
</tr>
<tr>
<td>_DATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

```plsql
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 2-22.

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.
This chapter describes how to implement APIs for:

- Project and task information
- Structure information
- Resource list and resource list member information
- Dependency information
- Task Assignment information

This chapter covers the following topics:

- Project Definition APIs
- Project Definition API Procedure Definitions
- Project Definition Record and Table Datatypes
- Using Project Definition APIs
- Creating a Project Using the Load-Execute-Fetch APIs
- Structure APIs
- Structure APIs Procedure Definitions
- User-Defined Attribute APIs
- Resource APIs
- Resource API Procedure Definitions
- Planning Resource List APIs
- Planning Resource List API Definitions
- Resource Breakdown Structure APIs
- Resource Breakdown Structure API Procedure Definitions
Project Definition 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 you have entered into your external system is unique in Oracle Projects
- Certain functions, such as deleting a project, 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.

Project Definition API Views

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

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_CUSTOMERS_LOV_V</td>
<td>Retrieves customers defined in or used by Oracle Projects</td>
</tr>
<tr>
<td>View</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_CLASS_CATEGORIES_LOV_V</td>
<td>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: &quot;select code description from pa_class_categories_lov_v where class_category = 'Funding Source';&quot;</td>
</tr>
<tr>
<td>PA_DISCOUNT_CODES_LOV_V</td>
<td>Retrieves the discount reasons specified for the lookup type RATE AND DISCOUNT REASON. Used to validate the parameters LABOR_DISC_REASON_CODE and NON_LABOR_DISC_REASON_CODE.</td>
</tr>
<tr>
<td>PA_DISTRIBUTION_RULES_LOV_V</td>
<td>Retrieves revenue distribution rules defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA_EMPLOYEE_SCHEDULES_LOV_V</td>
<td>Retrieves employee schedules. Only the following employee schedules are retrieved:</td>
</tr>
<tr>
<td></td>
<td>- If the MULTI_CURRENCY_BILLING_FLAG of the project is not Y, only schedules whose RATE_SCH_CURRENCY_CODE matches the project functional currency are retrieved.</td>
</tr>
<tr>
<td></td>
<td>- Only schedules whose SHARE_ACROSS_OU_FLAG is set to Y or whose operating unit matches the operating unit for the project are retrieved.</td>
</tr>
<tr>
<td>PA_JOB_SCHEDULES_LOV_V</td>
<td>Retrieves job schedules. Only the following job schedules are retrieved:</td>
</tr>
<tr>
<td></td>
<td>- If the MULTI_CURRENCY_BILLING_FLAG of the project is not Y, only schedules whose RATE_SCH_CURRENCY_CODE matches the project functional currency are retrieved.</td>
</tr>
<tr>
<td></td>
<td>- Only schedules whose SHARE_ACROSS_OU_FLAG is set to Y or whose operating unit matches the operating unit for the project are retrieved.</td>
</tr>
<tr>
<td></td>
<td>- Only schedules whose JOB_GROUP_ID matches the job group defined for the project type of the project are retrieved.</td>
</tr>
<tr>
<td>PA_INVOICE_SCHEDULES_LOV_V</td>
<td>Retrieves active invoice burden schedules</td>
</tr>
<tr>
<td>PA_KEY_MEMBERS_LOV_V</td>
<td>Retrieves names and employee identification numbers of team members from Oracle Projects. Note: pa_employees returns all employees defined in Oracle Projects.</td>
</tr>
</tbody>
</table>
View Description

PA_ORG_NL_SCHDL_LOV_V Retrieves non-labor schedules. Only the following non-labor schedules are retrieved:

- If the MULTI_CURRENCYBILLING_FLAG of the project is not Y, only schedules whose RATE_SCH_CURRENCY_CODE matches the project functional currency are retrieved.

- Only schedules whose SHARE_ACRROSS_OU_FLAG is set to Y or whose operating unit matches the operating unit for the project are retrieved.

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. This view provides a list of projects and its related attributes.

PA_REVENUE_SCHEDULES_LOV_V Retrieves active revenue burden schedules.

PA_SELECT_TEMPLATE_V Retrieves project templates and projects defined in Oracle Projects

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. The value of the ID field for all of the columns is 1020.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Display Name</th>
<th>Type</th>
<th>Order</th>
<th>Req?</th>
<th>View Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Project Name</td>
<td>20</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Name</td>
<td>Display Name</td>
<td>Type</td>
<td>Order</td>
<td>Req?</td>
<td>View Name</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------</td>
<td>---------------</td>
<td>-------</td>
<td>------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Project Description</td>
<td></td>
<td>30</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>START_DATE</td>
<td>Project Start Date</td>
<td></td>
<td>40</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>COMPLETION_DATE</td>
<td>Project Completion Date</td>
<td></td>
<td>50</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>PROJECT_STATUS</td>
<td>Project Status</td>
<td></td>
<td>60</td>
<td>N</td>
<td>PA_PROJECT_STATUS_LOV_V</td>
</tr>
<tr>
<td>PUBLIC_SECTOR</td>
<td>Public Sector</td>
<td></td>
<td>70</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>DISTRIBUTION_RULE</td>
<td>Distribution Rule</td>
<td></td>
<td>80</td>
<td>N</td>
<td>PA_DISTRIBUTION_RULES_LOV_V</td>
</tr>
<tr>
<td>CARRYING_OUT</td>
<td>Organization</td>
<td></td>
<td>90</td>
<td>N</td>
<td>PA_ORGANIZATIONS_LOV_V</td>
</tr>
<tr>
<td>KEY_MEMBER</td>
<td>Project Manager</td>
<td>PROJECT MANAGER</td>
<td>100</td>
<td>Y</td>
<td>PA_KEY_MEMBERS_LOV_V</td>
</tr>
<tr>
<td>KEY_MEMBER</td>
<td>Project Coordinator</td>
<td>Project Coordinator</td>
<td>110</td>
<td>N</td>
<td>PA_KEY_MEMBERS_LOV_V</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>Funding Source</td>
<td>Funding Source</td>
<td>120</td>
<td>Y</td>
<td>PA_CLASS_CATEGORIES_LOV_V</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>Market Sector</td>
<td>Market Sector</td>
<td>130</td>
<td>N</td>
<td>PA_CLASS_CATEGORIES_LOV_V</td>
</tr>
<tr>
<td>CUSTOMER_NAME</td>
<td>Customer Name</td>
<td>PRIMARY</td>
<td>140</td>
<td>N</td>
<td>PA_CUSTOMERS_LOV_V</td>
</tr>
</tbody>
</table>

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.
table below shows the valid values of the quick entry fields.

<table>
<thead>
<tr>
<th>Quick Entry Fields</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>(not validated)</td>
</tr>
<tr>
<td>CARRYING_OUT_ORGANIZATION_ID</td>
<td>PA_ORGANIZATIONS_LOV_V</td>
</tr>
<tr>
<td>PUBLIC_SECTOR_FLAG</td>
<td>Y or N</td>
</tr>
<tr>
<td>PROJECT_STATUS_CODE</td>
<td>PA_PROJECT_STATUS_LOV_V</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>(not validated)</td>
</tr>
<tr>
<td>START_DATE</td>
<td>DD-MON-YY format (e.g., 10-SEP-68)</td>
</tr>
<tr>
<td>COMPLETION_DATE</td>
<td>DD-MON-YY format (e.g., 13-JUL-94)</td>
</tr>
<tr>
<td>DISTRIBUTION_RULE</td>
<td>PA_DISTRIBUTION_RULES_LOV_V</td>
</tr>
<tr>
<td>CUSTOMER_ID</td>
<td>PA_CUSTOMERS_LOV_V (currently, the default CUSTOMER_RELATIONSHIP_CODE is PRIMARY. No other value is accepted)</td>
</tr>
<tr>
<td>KEY_MEMBERS (multiple)</td>
<td>PA_KEY_MEMBERS_LOV_V</td>
</tr>
<tr>
<td>CLASS_CATEGORIES (multiple)</td>
<td>PA_CLASS_CATEGORIES_LOV_V</td>
</tr>
</tbody>
</table>

**Project Definition API Procedures**

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

- Project Procedures
  - CREATE_PROJECT, page 3-8
  - DELETE_PROJECT, page 3-9
  - UPDATE_PROJECT, page 3-10

- Load-Execute-Fetch Procedures
- CLEAR_PROJECT, page 3-17
- EXECUTE_CREATE_PROJECT, page 3-17
- EXECUTE_UPDATE_PROJECT, page 3-18
- INIT_PROJECT, page 3-19
- LOAD_CLASSCATEGORY, page 3-19
- LOAD_KEY_MEMBER, page 3-20
- LOAD_ORG_ROLE, page 3-20
- LOAD_PROJECT, page 3-20

- Check Procedures
  - CHECK_CHANGE_PROJECT_ORG_OK, page 3-21
  - CHECK_DELETE_PROJECT_OK, page 3-22
  - CHECK_UNIQUE_PROJECT_REFERENCE, page 3-22

**Project Definition API Procedure Definitions**

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

**Common Project Definition API Parameters**

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

**PM_PROJECTREFERENCE**

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_PROJECTREFERENCE.
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 8-5.

### 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 if you are attempting to copy a project or template with tasks. CREATE_PROJECT also will not copy the WBS structure to the newly created project if you are attempting to copy a template which has the Automatically Publish Workplan Upon Project Creation option selected.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes, page 2-22.

**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.

### Business Rules

Oracle Projects imposes the following business rules.

### Performing Scheduling Validations

When set to 'N' the P_OP_VALIDATE_FLAG parameter eliminates redundant validation for certain types of scheduling data. Unnecessary scheduling validations can slow system performance.

You should set P_OP_VALIDATE_FLAG to 'N' if you are using this API to integrate a
third-party scheduling tool with Oracle Projects. Only set this parameter to 'N' if the third-party scheduling tool can perform validations for:

- Dependencies between tasks and activities
- Project schedule dates and task schedule dates

You should set P_OP_VALIDATE_FLAG to 'Y' if you are using the APIs to upload data from a legacy system that does not perform an extensive validation for the above. Scheduling data must be validated in order to ensure data integrity in Oracle Projects.

Parameters for CREATE_PROJECT

You can view descriptions of all of the parameters for CREATE_PROJECT in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The parameters for CREATE_PROJECT for which a value is required are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PROJECT_IN

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

You can view descriptions of all of the parameters for DELETE_PROJECT in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The parameters for DELETE_PROJECT for which a value is required are listed below:

• P_API_VERSION_NUMBER
• P_PM_PRODUCT_CODE

**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 2-22.

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 3-21 to identify the types of changes that Oracle Projects supports.

Update Mode

The Update Mode parameter has two possible values: PA_UPD_WBS_ATTR and PA_UPD_TASK_ATTR.

A value of PA_UPD_WBS_ATTR has the following effects:

• The workplan version is locked. (You cannot update the version if it is already locked by another user.)

• Users can update task information such as resource, dependencies, schedule and general details.

• Users can make updates to the workplan structure such as creating, deleting, increasing or decreasing indents, moving and copying tasks.

A value of PA_UPD_TASK_ATTR has the following effects:

• The workplan version remains unlocked.

• Users can update task information like dates, dependencies, resources and general details.

Business Rules (Project Level)

Oracle Projects imposes the following business rules.

Project Numbers, Project Names, Project Types, and Project Organizations

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:

<table>
<thead>
<tr>
<th>Entity or Topic</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project manager</td>
<td>A project can have only one active project manager.</td>
</tr>
<tr>
<td>New project manager</td>
<td>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.</td>
</tr>
<tr>
<td>Team members</td>
<td>A project can have any number of team members other than the project manager.</td>
</tr>
<tr>
<td>Team member start date</td>
<td>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. It is mandatory for UPDATE_PROJECT that key members have a start date, however. You cannot directly update the start date of an existing key member. To specify a start date for an existing key member, you need to end date the key member with a date prior to the new start date. You can then create a new entry for the key member with a new start date.</td>
</tr>
<tr>
<td>Team roles during different periods</td>
<td>UPDATE_PROJECT supports a person performing the same role (other than project manager) for a given project during different periods.</td>
</tr>
<tr>
<td>Primary customer</td>
<td>A project can have only one primary customer.</td>
</tr>
<tr>
<td>Bill to address ID</td>
<td>Customers can update the bill to address ID of the current bill to customer by passing a valid value for the address. In CREATE_PROJECT or when using UPDATE_PROJECT to create a new customer, if no value is passed for the bill to address, then the primary address for the customer is used.</td>
</tr>
<tr>
<td>Ship to address ID</td>
<td>Customers can update the ship to address ID of the current bill to customer by passing a valid value for the address. In CREATE_PROJECT or when using UPDATE_PROJECT to create a new customer, if no value is passed for the ship to address, then the primary address for the customer is used.</td>
</tr>
</tbody>
</table>
### Rules for Project Start and End Dates

The rules for project start and end dates are listed below:

- **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.)

**Performing Scheduling Validations**

When set to 'N' the P_OP_VALIDATE_FLAG parameter eliminates redundant validation for certain types of scheduling data. Unnecessary scheduling validations can slow system performance.

You should set P_OP_VALIDATE_FLAG to 'N' if you are using this API to integrate a third-party scheduling tool with Oracle Projects. Only set this parameter to 'N' if the third-party scheduling tool can perform validations for:

- Dependencies between tasks and activities
- Project schedule dates and task schedule dates

You should set P_OP_VALIDATE_FLAG to 'Y' if you are using the APIs to upload data from a legacy system that does not perform an extensive validation for the above. Scheduling data must be validated in order to ensure data integrity in Oracle Projects.

**Rules for Updating Labor and Non-Labor Billing and Burdening Schedule Attributes for a Project**

The rules for updating labor and non-labor schedule attributes are listed below:

- If the labor schedule type is being changed from Bill to Burden or from Burden to Bill, then the column LABOR_SCH_TYPE is required, and all the required dependent columns must be populated.

- If the labor or non-labor schedule type is being changed from Bill to Burden, then the columns REV_IND_RATE_SCH_ID and INV_IND_RATE_SCH_ID are required.

- If the labor schedule type is being changed from Burden to Bill, and Oracle Project Resource Management is installed, then the column JOB_BILL_RATE_SCHEDULE_ID is required.

- If the non-labor schedule type is being changed from Burden to Bill, then the columns NON_LABOR_BILL_RATE_ORG_ID and NON_LAB_STD_BILL_RT_SCH_ID are required.

- If the LABOR_SCHEDULE_DISCOUNT is passed, then the LABOR_SCHEDULE_DISCOUNT_REASON is also required.

- If the LABOR_SCHEDULE_DISCOUNT is not passed, then the LABOR_SCHEDULE_DISCOUNT_REASON cannot be passed.
• If the NON_LABOR_SCHEDULE_DISCOUNT is passed, then the NON_LABOR_SCHEDULE_DISCOUNT_REASON is also required.

• If the NON_LABOR_SCHEDULE_DISCOUNT is not passed, then the NON_LABOR_SCHEDULE_DISCOUNT_REASON cannot be passed.

• Any schedule ID or organization ID that is passed must be a valid ID.

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
The 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

Updating Labor and Non-Labor Billing and Burdening Schedule Attributes

The rules for updating labor and non-labor schedule attributes for tasks are the same as the corresponding rules for projects. See: Rules for Updating Labor and Non-Labor Billing and Burdening Schedule Attributes for a Project, page 3-14

Parameters for UPDATE_PROJECT

You can view descriptions of all of the parameters for UPDATE_PROJECT in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The parameters for which a value is required for UPDATE_PROJECT are listed below:

• P_API_VERSION_NUMBER
• P_PM_PRODUCT_CODE
• P_PROJECT_IN

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 3-85.

**Business Rules**

Oracle Projects imposes the following business rules.

**Performing Scheduling Validations**

When set to 'N' the P_OP_VALIDATE_FLAG parameter eliminates redundant validation for certain types of scheduling data. Unnecessary scheduling validations can slow system performance.

You should set P_OP_VALIDATE_FLAG to 'N' if you are using this API to integrate a third-party scheduling tool with Oracle Projects. Only set this parameter to 'N' if the third-party scheduling tool can perform validations for:

- Dependencies between tasks and activities
- Project schedule dates and task schedule dates

You should set P_OP_VALIDATE_FLAG to 'Y' if you are using the APIs to upload data from a legacy system that does not perform an extensive validation for the above. Scheduling data must be validated in order to ensure data integrity in Oracle Projects.

**Parameters for EXECUTE_CREATE_PROJECT**

You can view descriptions of all of the parameters for EXECUTE_CREATE_PROJECT in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for EXECUTE_CREATE_PROJECT are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

**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 3-85.
Business Rules

Oracle Projects imposes the following business rules.

Performing Scheduling Validations

When set to ‘N’ the P_OP_VALIDATE_FLAG parameter eliminates redundant validation for certain types of scheduling data. Unnecessary scheduling validations can slow system performance.

You should set P_OP_VALIDATE_FLAG to ‘N’ if you are using this API to integrate a third-party scheduling tool with Oracle Projects. Only set this parameter to ‘N’ if the third-party scheduling tool can perform validations for:

- Dependencies between tasks and activities
- Project schedule dates and task schedule dates

You should set P_OP_VALIDATE_FLAG to ‘Y’ if you are using the APIs to upload data from a legacy system that does not perform an extensive validation for the above. Scheduling data must be validated in order to ensure data integrity in Oracle Projects.

Parameters for EXECUTE_UPDATE_PROJECT

You can view descriptions of all of the parameters for EXECUTE_UPDATE_PROJECT in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for EXECUTE_UPDATE_PROJECT are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

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.

You can view descriptions of all of the parameters for LOAD_CLASS_CATEGORY in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for LOAD_CLASS_CATEGORY are listed below:
• P_API_VERSION_NUMBER
• P_CLASSCATEGORY (depending on template setup)
• P_CLASSCODE (if P_CLASSCATEGORY is not NULL)

LOAD_KEY_MEMBER

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

You can view descriptions of all of the parameters for LOAD_KEY_MEMBER in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for LOAD_KEY_MEMBER are listed below:
• P_API_VERSION_NUMBER
• P_PERSON_ID (depending on template setup)
• P_PROJECT_ROLE_TYPE (if P_PERSON_ID is not NULL)

LOAD_ORG_ROLE

LOAD_ORG_ROLE is a Load-Execute-Fetch procedure used to load organization roles from the client side to a PL/SQL table on the server side, where the roles will be used by the Load-Execute-Fetch cycle.

You can view descriptions of all of the parameters for LOAD_ORG_ROLE in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for LOAD_ORG_ROLE are listed below:
• P_API_VERSION_NUMBER
• P_RESOURCE_SOURCE_ID (depending on template setup)
• P_PROJECT_ROLE_TYPE (if P_RESOURCE_SOURCE_ID is not NULL)

LOAD_PROJECT

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

You can view descriptions of all of the parameters for LOAD_PROJECT in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for LOAD_PROJECT are listed below:

- P_API_VERSION_NUMBER
- P_PM_PROJECT_REFERENCE
- P_PROJECT_NAME
- P_CREATED_FROM_PROJECT_ID

The following parameters may be required, depending on the setup of the project template:

- P_CARRYING_OUT_ORGANIZATION_ID
- P_PUBLIC_SECTOR_FLAG
- P_PROJECT_STATUS_CODE
- P_DESCRIPTION
- P_START_DATE
- P_COMPLETION_DATE
- P_DISTRIBUTION_RULE
- P_CUSTOMER_ID
- P_PROJECT_RELATIONSHIP_CODE

Check Procedures

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

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

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.

You can view descriptions of all of the parameters for CHECK_CHANGE_PROJECT_ORG_OK in the Oracle Integration Repository. The
Oracle Integration Repository is described in the preface of this manual.

The required parameters for CHECK_CHANGE_PROJECT_ORG_OK are listed below:
- P_API_VERSION_NUMBER

**CHECK_DELETE_PROJECT_OK**

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

You can view descriptions of all of the parameters for CHECK_DELETE_PROJECT_OK in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for CHECK_DELETE_PROJECT_OK are listed below:
- P_API_VERSION_NUMBER

**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.

You can view descriptions of all of the parameters for CHECK_UNIQUE_PROJECT_REFERENCE in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for CHECK_UNIQUE_PROJECT_REFERENCE are listed below:
- P_API_VERSION_NUMBER

**Project Definition 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_PROJECT_REFERENCE</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>Code for the project in the external system.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_PROJECT_ID</td>
<td>NUMBER (15)</td>
<td>For update</td>
<td>The reference code that uniquely identifies the project in Oracle Projects</td>
</tr>
<tr>
<td>PA_PROJECT_NUMBER</td>
<td>VARCHAR2 (25)</td>
<td>No</td>
<td>The project number that uniquely identifies the project in Oracle Projects</td>
</tr>
<tr>
<td>PROJECT_NAME</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>Unique name of the project uniquely identifies the project in Oracle Projects</td>
</tr>
<tr>
<td>LONG_NAME</td>
<td>VARCHAR2 (240)</td>
<td>No</td>
<td>Project long name</td>
</tr>
<tr>
<td>CREATED_FROM_PROJECT_ID</td>
<td>NUMBER (15)</td>
<td>Yes</td>
<td>Number that uniquely identifies the template from which this project originates</td>
</tr>
<tr>
<td>CARRYING_OUT_ORGANIZATION_ID</td>
<td>NUMBER (15)</td>
<td>Based on template setup</td>
<td>The identification code of the organization responsible for the project work</td>
</tr>
<tr>
<td>PUBLIC_SECTOR_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>Based on template setup</td>
<td>Flag that indicates whether this project is in the Public or the Private sector</td>
</tr>
<tr>
<td>PROJECT_STATUS_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Based on template setup</td>
<td>The status of the project. Any status other than CLOSED is considered active.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2 (250)</td>
<td>Based on template setup</td>
<td>The description of the project</td>
</tr>
<tr>
<td>START_DATE</td>
<td>DATE</td>
<td>Based on template setup</td>
<td>The date on which the project starts</td>
</tr>
<tr>
<td>COMPLETION_DATE</td>
<td>DATE</td>
<td>Based on template setup</td>
<td>The date on which the project is completed</td>
</tr>
<tr>
<td>DISTRIBUTION_RULE</td>
<td>VARCHAR2 (30)</td>
<td>Based on template setup</td>
<td>The distribution rule that specifies the contract project’s revenue accrual and billing method</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CUSTOMER_ID</td>
<td>NUMBER</td>
<td>Based on template setup</td>
<td>The identification code of the project's customer</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_RELATIONSHIP_CODE</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>The type of customer relationship the customer has on the project</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTUAL_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The actual project start date in the external system</td>
</tr>
<tr>
<td>ACTUAL_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The actual project finish date in the external system</td>
</tr>
<tr>
<td>EARLY_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The early project start date in the external system</td>
</tr>
<tr>
<td>EARLY_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The early project finish date in the external system</td>
</tr>
<tr>
<td>LATE_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The late project start date in the external system</td>
</tr>
<tr>
<td>LATE_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The late project finish date in the external system</td>
</tr>
<tr>
<td>SCHEDULED_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled project start date in the external system</td>
</tr>
<tr>
<td>SCHEDULED_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled project finish date in the external system</td>
</tr>
<tr>
<td>ATTRIBUTE_CATEGORY</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Used by descriptive flexfields</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE1 THROUGH ATTRIBUTE10</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Descriptive flexfield</td>
</tr>
<tr>
<td></td>
<td>(150)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| OUTPUT_TAX_CODE           | VARCHAR2       | No        | Indicates whether tax rate defined for the Project will be used for Customer Invoices.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETENTION_TAX_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Indicates whether tax rate defined for the Retention will be used for Customer Invoices.</td>
</tr>
<tr>
<td>PROJECT_CURRENCY_CODE</td>
<td>VARCHAR2 (15)</td>
<td>No</td>
<td>Project currency code. The default value is the currency code of the ledger.</td>
</tr>
<tr>
<td>ALLOW_CROSS_CHARGE_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Cross charge allowed? Value is required. Default Value is N. This value can be overridden at any task level.</td>
</tr>
<tr>
<td>PROJECT_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Default project currency rate date (date for accounting currency rate for a given rate type).</td>
</tr>
<tr>
<td>PROJECT_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Default project currency rate type (e.g., Spot, Corporate).</td>
</tr>
<tr>
<td>CC_PROCESS_LABOR_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>LABOR_TP_SCHEDULE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>LABOR_TPFIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>CC_PROCESS_NL_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NL_TP_SCHEDULE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>NL_TP_FIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>CC_TAX_TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task to which intercompany tax items on the intercompany AP invoice are charged.</td>
</tr>
<tr>
<td>ROLE_LIST_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the role list, a list of allowable roles that are displayed when team members are assigned</td>
</tr>
<tr>
<td>WORK_TYPE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Work type identifier. Work types are predefined types of work. For example, Vacation, Training, and Administration.</td>
</tr>
<tr>
<td>CALENDAR_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Calendar identifier. A calendar specifies exceptions such as public holidays.</td>
</tr>
<tr>
<td>LOCATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the project work site location</td>
</tr>
<tr>
<td>PROBABILITY_MEMBER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the probability member. Project probability, the likelihood that a project will be approved, is used as a weighting average for reporting.</td>
</tr>
<tr>
<td>PROJECT_VALUE</td>
<td>NUMBER</td>
<td>No</td>
<td>The opportunity value converted to the project functional currency</td>
</tr>
<tr>
<td>EXPECTED_APPROVAL_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The expected date of the project approval (for information purposes only)</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COST_JOB_GROUP_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the job group for costing functionality</td>
</tr>
<tr>
<td>BILL_JOB_GROUP_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the job group for billing functionality</td>
</tr>
<tr>
<td>TEAM_TEMPLATE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The team template that you want to add to a new project</td>
</tr>
<tr>
<td>TEAM_TEMPLATE_ID</td>
<td>NUMBER</td>
<td>(15)</td>
<td></td>
</tr>
<tr>
<td>COUNTRY_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Country name code</td>
</tr>
<tr>
<td>COUNTRY_CODE</td>
<td>VARCHAR2</td>
<td>(250)</td>
<td></td>
</tr>
<tr>
<td>REGION</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Region</td>
</tr>
<tr>
<td>CITY</td>
<td>VARCHAR2</td>
<td>No</td>
<td>City</td>
</tr>
<tr>
<td>CITY</td>
<td>VARCHAR2</td>
<td>(250)</td>
<td></td>
</tr>
<tr>
<td>EMP_BILL_RATE_SCHEDULE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the employee-based bill rate schedule for the project</td>
</tr>
<tr>
<td>JOB_BILL_RATE_SCHEDULE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the job-based bill rate schedule for the project</td>
</tr>
<tr>
<td>INVPROC_CURRENCY_TYPE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Invoice processing currency code</td>
</tr>
<tr>
<td>INVPROC_CURRENCY_TYPE</td>
<td>VARCHAR2</td>
<td>(30)</td>
<td></td>
</tr>
<tr>
<td>REVPROC_CURRENCY_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Revenue processing currency code in the project functional currency</td>
</tr>
<tr>
<td>REVPROC_CURRENCY_CODE</td>
<td>VARCHAR2</td>
<td>(15)</td>
<td></td>
</tr>
<tr>
<td>PROJECT_BIL_RATE_DATE_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Exchange rate date type for converting customer billing amounts from bill</td>
</tr>
<tr>
<td>PROJECT_BIL_RATE_DATE_CODE</td>
<td>VARCHAR2</td>
<td>(30)</td>
<td>transaction currency or funding currency to project currency</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROJECT_BIL_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Exchange rate type for converting customer billing amounts from bill transaction currency or funding currency to project currency</td>
</tr>
<tr>
<td>PROJECT_BIL_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>PROJECT_BIL_EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Exchange rate for conversion from bill transaction currency or funding currency to project currency if the rate type is User</td>
</tr>
<tr>
<td>PROJFUNC_CURRENCY_CODE</td>
<td>VARCHAR2 (15)</td>
<td>No</td>
<td>Project functional currency. The default value is the value entered for the associated ledger.</td>
</tr>
<tr>
<td>PROJFUNC_BIL_RATE_DATE_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Exchange rate date type for converting customer billing amounts from bill transaction currency or funding currency to project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_BIL_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Exchange rate type for converting customer billing amounts from bill transaction currency or funding currency to project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_BIL_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>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</td>
</tr>
<tr>
<td>PROJFUNC_BIL_EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Exchange rate for conversion from bill transaction currency or funding currency to project functional currency if the rate type is User</td>
</tr>
<tr>
<td>FUNDING_RATE_DATE_Code</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Exchange rate date type for converting customer billing amounts from bill transaction currency to funding currency</td>
</tr>
<tr>
<td>FUNDING_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Exchange rate type for converting customer billing amounts from bill transaction currency to funding currency</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FUNDING_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Exchange rate date for converting customer billing amounts from bill transaction currency to funding currency if rate date type is Fixed</td>
</tr>
<tr>
<td>FUNDING_EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Exchange rate for conversion from bill transaction currency to project or functional currency if rate type is User</td>
</tr>
<tr>
<td>BASELINE_FUNDING_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether the funding can be baselined without a revenue budget</td>
</tr>
<tr>
<td>MULTI_CURRENCY_BILLING_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether multi-currency billing is allowed for the project</td>
</tr>
<tr>
<td>COMPETENCE_MATCH_WT</td>
<td>NUMBER</td>
<td>No</td>
<td>The weighting value for competence match, used to calculate the score</td>
</tr>
<tr>
<td>AVAILABILITY_MATCH_WT</td>
<td>NUMBER</td>
<td>No</td>
<td>The weighting value for availability match, used to calculate the score</td>
</tr>
<tr>
<td>JOB_LEVEL_MATCH_WT</td>
<td>NUMBER</td>
<td>No</td>
<td>The weighting value for job-level match, used to calculate the score</td>
</tr>
<tr>
<td>ENABLE_AUTOMATED_SEARCH</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether automated candidate nomination is used for the requirements on a project</td>
</tr>
<tr>
<td>SEARCH_MIN_AVAILABILITY</td>
<td>NUMBER</td>
<td>No</td>
<td>The minimum required availability for a resource to be returned in the search result</td>
</tr>
<tr>
<td>SEARCH_ORG_HIER_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>Organization hierarchy for searches</td>
</tr>
<tr>
<td>SEARCH_STARTING_ORG_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>Starting organization for searches</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SEARCH_COUNTRY_ID</td>
<td>VARCHAR2 (2)</td>
<td>No</td>
<td>Country for the search</td>
</tr>
<tr>
<td>MIN_CAND_SCORE_REQD_FOR_NOM</td>
<td>NUMBER</td>
<td>No</td>
<td>Minimum score required for a resource to be nominated as candidate on a requirement</td>
</tr>
<tr>
<td>MAX_NUM_OF_SYS_NOM_CAND</td>
<td>NUMBER</td>
<td>No</td>
<td>Maximum number of candidates that can be nominated</td>
</tr>
<tr>
<td>NON_LAB_STD_BILL_RT_SCH_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>Identifier of the non-labor standard bill rate schedule</td>
</tr>
<tr>
<td>SEARCH_COUNTRY_CODE</td>
<td>VARCHAR2 (2)</td>
<td>No</td>
<td>Country for searches</td>
</tr>
<tr>
<td>INV_BY_BILL_TRANS_CURR_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether invoicing is by bill transaction currency for the project</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Default value for the project functional cost rate</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Default value for the project functional cost rate date</td>
</tr>
<tr>
<td>ASSIGN_PRECEDES_TASK</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether assignment level attributes override task level attributes</td>
</tr>
<tr>
<td>SPLIT_COST_FROM_WORKPLAN_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag indicating whether split cost from workplan is permitted</td>
</tr>
<tr>
<td>SPLIT_COST_FROM_BILL_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag indicating whether split cost from bill is permitted</td>
</tr>
<tr>
<td>ADV_ACTION_SET_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>Flag that indicates the default advertisement action set of the project or project template</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>START_ADV_ACTION_SET_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether the advertisement action set will start immediately after a requirement is created</td>
</tr>
<tr>
<td>PRIORITY_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The code identifying the priority of the project</td>
</tr>
<tr>
<td>RETN_BILLING_INV_FORMAT_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>The identifier of the retention billing invoice format</td>
</tr>
<tr>
<td>RETN_ACCOUNTING_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether retention accounting is enabled for the project</td>
</tr>
<tr>
<td>OPP_VALUE_CURRENCY_CODE</td>
<td>VARCHAR2 (15)</td>
<td>No</td>
<td>The currency code for project opportunity value</td>
</tr>
<tr>
<td>REvaluate_Funding_Flag</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether the funding has to be reevaluated</td>
</tr>
<tr>
<td>INCLUDE_GAINS_LOSSES_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether gains and losses to be included in project revenue</td>
</tr>
<tr>
<td>SECURITY_LEVEL</td>
<td>NUMBER</td>
<td>No</td>
<td>Indicator showing whether a project is public or private. A value of 101 (secured) indicates that the project is private. A value of 1 (enterprise) indicates that the project is public.</td>
</tr>
<tr>
<td>LABOR_DISC_REASON_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Reason code for labor discount</td>
</tr>
<tr>
<td>NON_LABOR_DISC_REASON_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Reason code for non-labor discount</td>
</tr>
<tr>
<td>LABOR_SCHEDULE_FIXED_DATE</td>
<td>DATE</td>
<td>For update</td>
<td>The date used to determine the effective bill rates when using the labor bill rate schedule</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LABOR_SCHEDULE_DISCOUNT</td>
<td>DATE</td>
<td>For update</td>
<td>The discount for the labor bill rate schedule</td>
</tr>
<tr>
<td>NON_LABOR_BILL_RATE_ORG_ID</td>
<td>NUMBER</td>
<td>For update</td>
<td>The identifier of the organization attached to the non-labor bill rate schedule</td>
</tr>
<tr>
<td>NON_LABOR_SCHEDULE_FIXED_DATE</td>
<td>DATE</td>
<td>For update</td>
<td>The date when the schedule will become effective</td>
</tr>
<tr>
<td>NON_LABOR_SCHEDULE_DISCOUNT</td>
<td>DATE</td>
<td>For update</td>
<td>The discount for the non-labor bill rate schedule</td>
</tr>
<tr>
<td>REV_IND_RATE_SCH_ID</td>
<td>NUMBER</td>
<td>For update</td>
<td>The identifier of the revenue schedule</td>
</tr>
<tr>
<td>INV_IND_RATE_SCH_ID</td>
<td>NUMBER</td>
<td>For update</td>
<td>The identifier of the invoice schedule</td>
</tr>
<tr>
<td>REV_IND_SCH_FIXED_DATE</td>
<td>DATE</td>
<td>For update</td>
<td>The start date of the revenue schedule for a burden schedule of the type Firm</td>
</tr>
<tr>
<td>INV_IND_SCH_FIXED_DATE</td>
<td>DATE</td>
<td>For update</td>
<td>The start date of the invoice schedule for a burden schedule of the type Firm</td>
</tr>
<tr>
<td>LABOR_SCH_TYPE</td>
<td>VARCHAR (1)</td>
<td>For update</td>
<td>The schedule type (Burden or Bill) for labor expenditure items</td>
</tr>
<tr>
<td>NON_LABOR_SCH_TYPE</td>
<td>VARCHAR (1)</td>
<td>For update</td>
<td>The schedule type (Burden or Bill) for non-labor expenditure items</td>
</tr>
<tr>
<td>ASSET_ALLOCATION_METHOD</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The method used to allocate indirect and common costs across the assets assigned to a grouping level</td>
</tr>
<tr>
<td>CAPITAL_EVENT_PROCESSING</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The capital event processing method, used to determine when cost and assets are grouped for capitalization or retirement adjustment processing</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>CINT_RATE_SCH_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the capital interest rate schedule</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CINT_ELIGIBLE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates whether the project is eligible for capitalized interest</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CINT_STOP_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Stop date for capital interest calculation</td>
</tr>
<tr>
<td>BILL_TO_CUSTOMER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the bill to customer name</td>
</tr>
<tr>
<td>SHIP_TO_CUSTOMER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the ship to customer name</td>
</tr>
<tr>
<td>PROCESS_MODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Processing mode. Indicates whether task processing should be done online or using a concurrent request</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYS_PROGRAM_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag indicating whether the project can be treated as a program</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENABLE_TOP_TASK_CUSTOMER_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag indicating whether customer at top task is enabled for the project</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENABLE_TOP_TASK_INV_MTH_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag indicating whether the invoice method at top task is enabled for the project</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJFUNC_ATTR_FOR_AR_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BILL_TO_ADDRESS_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the bill to customer address</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIP_TO_ADDRESS_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the ship to customer address</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PROJECT_OUT_REC_TYPE Datatype**

The following table shows the PROJECT_OUT_REC_TYPE datatype.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_PROJECT_ID</td>
<td>NUMBER</td>
<td>The reference code that uniquely identifies the project in Oracle Projects (15)</td>
</tr>
<tr>
<td>PA_PROJECT_NUMBER</td>
<td>VARCHAR2</td>
<td>The number that uniquely identifies the project in Oracle Projects (25)</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>VARCHAR2</td>
<td>API standard</td>
</tr>
</tbody>
</table>

**PROJECT_ROLE_TBL_TYPE Datatype**

The following table shows the PROJECT_ROLE_TBL_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON_ID</td>
<td>NUMBER</td>
<td>Based on template setup</td>
<td>The identifier of the person (9)</td>
</tr>
<tr>
<td>PROJECT_ROLE_TYPE</td>
<td>VARCHAR2</td>
<td>Yes, if PERSON_ID is not NULL</td>
<td>The type of role that the person has on the project (20)</td>
</tr>
<tr>
<td>START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The date when the role begins for the person. Default value is the project start date.</td>
</tr>
<tr>
<td>END_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The date when the role ends for the person</td>
</tr>
</tbody>
</table>

**CLASSCATEGORY_TBL_TYPE Datatype**

The following table shows the CLASSCATEGORY_TBL_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASSCATEGORY</td>
<td>VARCHAR2</td>
<td>Based on template setup</td>
<td>The class category of the project (30)</td>
</tr>
</tbody>
</table>
### Oracle Project Foundation APIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Yes, if CLASS_CATEGORY is not NULL</td>
<td>The class code of the project</td>
</tr>
<tr>
<td>CODE_PERCENTAGE</td>
<td>NUMBER</td>
<td>No</td>
<td>Class category percentage</td>
</tr>
<tr>
<td>NEW_CLASS_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The new class code to replace the existing CLASS_CODE. If the project is not already assigned a CLASS_CODE, this parameter is ignored.</td>
</tr>
</tbody>
</table>

#### 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_TASK_REFERENCE</td>
<td>VARCHAR2 (25)</td>
<td>Yes, or PA_TASK_ID is used</td>
<td>The identifier of the task in the external system</td>
</tr>
<tr>
<td>PA_TASK_ID</td>
<td>NUMBER (15)</td>
<td>For update</td>
<td>The reference code that uniquely identifies a task within a project in Oracle Projects</td>
</tr>
<tr>
<td>TASK_NAME</td>
<td>VARCHAR2 (20)</td>
<td>Yes</td>
<td>The name that uniquely identifies a task within a project</td>
</tr>
<tr>
<td>PA_TASK_NUMBER</td>
<td>VARCHAR2 (25)</td>
<td>Yes</td>
<td>The number that identifies the task in Oracle Projects. Intended for systems that maintain a task number in addition to a unique task reference.</td>
</tr>
<tr>
<td>TASK_DESCRIPTION</td>
<td>VARCHAR2 (250)</td>
<td>No</td>
<td>Description of the task</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TASK_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The date on which the task starts</td>
</tr>
<tr>
<td>TASK_COMPLETION_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The date on which the task is completed</td>
</tr>
<tr>
<td>PM_PARENT_TASK_REFERENCE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The reference code that identifies the task's parent task in the external system</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PA_PARENT_TASK_ID</td>
<td>NUMBER</td>
<td>For update</td>
<td>The identification code of the task's parent task in Oracle Projects</td>
</tr>
<tr>
<td>ADDRESS_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The address of one of the customers logically linked to this task</td>
</tr>
<tr>
<td>CARRYING_OUT_ORGANIZATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identification code of the organization responsible for the task work. The task organization defaults to the project organization upon creation of the task.</td>
</tr>
<tr>
<td>SERVICE_TYPE_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The type of work performed on the task</td>
</tr>
<tr>
<td>TASK_MANAGER_PERSON_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>BILLABLE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Default flag for items charged to the task that indicates if the item can accrue revenue (Y or N)</td>
</tr>
<tr>
<td>CHARGEABLE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates if expenditure items can be charged to the task. Only lowest tasks are chargeable.</td>
</tr>
<tr>
<td>READY_TO_BILL_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates whether the task is authorized to be invoiced</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>READY_TO_DISTRIBUTE_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether the task is authorized for revenue accrual</td>
</tr>
<tr>
<td>LIMIT_TO_TXN_CONTROLS_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates that users can charge to the task only those expenditures listed in the task’s transaction controls</td>
</tr>
<tr>
<td>LABOR_BILL_RATE_ORG_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>The identification code of the organization that owns the labor standard bill rate schedule</td>
</tr>
<tr>
<td>LABOR_STD_BILL_RATE_SCHDL</td>
<td>VARCHAR2 (20)</td>
<td>No</td>
<td>The labor standard bill rate schedule used to calculate revenue for labor expenditure items charged to the task</td>
</tr>
<tr>
<td>LABOR_SCHEDULE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The date used to determine the effective bill rates of the task standard labor bill rate schedule</td>
</tr>
<tr>
<td>LABOR_SCHEDULE_DISCOUNT</td>
<td>NUMBER (7,4)</td>
<td>No</td>
<td>The percentage to be discounted from the task standard labor bill rate schedule</td>
</tr>
<tr>
<td>NON_LABOR_BILL_RATE_ORG_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>The identification code of the organization that owns the non-labor standard bill rate schedule</td>
</tr>
<tr>
<td>NON_LABOR_STD_BILL_RATE_SCHDL</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The non-labor standard bill rate schedule used to calculate revenue for non-labor expenditure items charged to the task</td>
</tr>
<tr>
<td>NON_LABOR_SCHEDULE_FIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The fixed date used to determine the effective bill rates of the standard non-labor bill rate schedule</td>
</tr>
<tr>
<td>NON_LABOR_SCHEDULE_DISCOUNT</td>
<td>NUMBER (7,4)</td>
<td>No</td>
<td>The percentage to be discounted from the task standard non-labor bill rate schedule</td>
</tr>
<tr>
<td>LABOR_COST_MULTIPLIER_NAME</td>
<td>VARCHAR2 (20)</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>COST_IND_RATE_SCH_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>The identification code of the default costing burden schedule</td>
</tr>
<tr>
<td>REV_IND_RATE_SCH_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>The identification code of the default revenue burden schedule</td>
</tr>
<tr>
<td>INV_IND_RATE_SCH_ID</td>
<td>NUMBER (15)</td>
<td>No</td>
<td>The identification code of the default invoice burden schedule</td>
</tr>
<tr>
<td>COST_IND_SCH_FIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled fixed date of the firm costing burden schedule</td>
</tr>
<tr>
<td>REV_IND_SCH_FIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled fixed date of the firm revenue burden schedule</td>
</tr>
<tr>
<td>INV_IND_SCH_FIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled fixed date of the firm invoice burden schedule</td>
</tr>
<tr>
<td>LABOR_SCH_TYPE</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>The scheduled type of labor expenditure items</td>
</tr>
<tr>
<td>NON_LABOR_SCH_TYPE</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>The scheduled type of non-labor expenditure items</td>
</tr>
<tr>
<td>ACTUAL_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The actual start date of the project in the external system</td>
</tr>
<tr>
<td>ACTUAL_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The actual finish date of the project in the external system</td>
</tr>
<tr>
<td>EARLY_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The early start date of the project in the external system</td>
</tr>
<tr>
<td>EARLY_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The early finish date of the project in the external system</td>
</tr>
<tr>
<td>LATE_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The late start date of the project in the external system</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LATE_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The late finish date of the project in the external system</td>
</tr>
<tr>
<td>SCHEDULED_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled start date of the project in the external system</td>
</tr>
<tr>
<td>SCHEDULED_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The scheduled finish date of the project in the external system</td>
</tr>
<tr>
<td>ALLOW_CROSS_CHARGE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Cross charge allowed? Value is required. Default Value is N. This value can be overridden at any task level.</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Default project currency rate date (date for accounting currency rate for a given rate type).</td>
</tr>
<tr>
<td>PROJECT_RATE_TYPE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Default project currency rate type (e.g., Spot, Corporate).</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC_PROCESS_LABOR_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LABOR_TP_SCHEDULE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>LABOR_TP_FIXED_DATE</td>
<td>DATE</td>
<td>No</td>
<td>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_NL_FLAG is set to Y, this field is required.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CC_PROCESS_NL_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicated cross charge processing is to be performed for non-labor transactions charged to the project. Default value for the project template is N. This is defaulted to a project from the project template.</td>
</tr>
<tr>
<td>NL_TP_SCHEDULE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier for transfer price schedule for cross charged non-transactions. This is defaulted to a project from the project template. If CC_PROCESS_NL_FLAG is set to Y, this field is required.</td>
</tr>
<tr>
<td>NL_TP_FIXED_DAT</td>
<td>DATE</td>
<td>No</td>
<td>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.</td>
</tr>
<tr>
<td>RECEIVE_PROJECT</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates that the task may receive charges from internal suppliers via inter-project billing.</td>
</tr>
<tr>
<td>ATTRIBUTE_CATEGORY</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Used by descriptive flexfields</td>
</tr>
<tr>
<td>ATTRIBUTE1 THROUGH</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Descriptive flexfield</td>
</tr>
<tr>
<td>ATTRIBUTE10</td>
<td>(150)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_JOB_BILL_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the job-based bill rate schedule for the project</td>
</tr>
<tr>
<td>P_EMP_BILL_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the employee-based bill rate schedule for the project</td>
</tr>
<tr>
<td>P_TASKFUNC_COST</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The task-level default value for project functional cost rate type</td>
</tr>
<tr>
<td>P_TASKFUNC_COST</td>
<td>DATE</td>
<td>No</td>
<td>The task-level default value for project functional cost rate date</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>----------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>P_NON_LAB_STD_BILL_RT_SCH_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the non-labor standard bill rate schedule</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LABOR_DISC_REASON_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Reason code for labor discount</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_NON_LABOR_DISC_REASON_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Reason code for non-labor discount</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_LONG_TASK_NAME</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Task long name</td>
</tr>
<tr>
<td></td>
<td>(240)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P RETIREMENT_COST_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that identifies tasks for retirement cost collection</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_CINT_ELIGIBLE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates whether the project is eligible for</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td>capitalized interest</td>
</tr>
<tr>
<td>P_CINT_STOP_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Stop date for capital interest calculation</td>
</tr>
<tr>
<td>P_REVENUE_ACCRUAL_METHOD</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The revenue accrual method for task</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_INVOICE_METHOD</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The invoice method for the task</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_OBLIGATION_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The obligation start date of the workplan version</td>
</tr>
<tr>
<td>P_OBLIGATION_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The obligation finish date of the workplan version</td>
</tr>
<tr>
<td>P_ACTUAL_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The actual start date of the workplan version</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>----------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_ACTUAL_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The actual end date of the workplan version</td>
</tr>
<tr>
<td>P_ESTIMATED_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The estimated start date of the workplan version</td>
</tr>
<tr>
<td>P_ESTIMATED_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The estimated finish date of the workplan version</td>
</tr>
<tr>
<td>P_EARLY_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The early start date of the workplan version</td>
</tr>
<tr>
<td>P_EARLY_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The early finish date of the workplan version</td>
</tr>
<tr>
<td>P_LATE_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The late start date of the workplan version</td>
</tr>
<tr>
<td>P_LATE_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The late finish date of the workplan version</td>
</tr>
<tr>
<td>P_MILESTONE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates if the task version is a milestone. This is a task-specific attribute.</td>
</tr>
<tr>
<td>P_CRITICAL_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag that indicates if the task version is part of the critical path. This is a task-specific attribute.</td>
</tr>
<tr>
<td>P_WQ_PLANNED_QUANTITY</td>
<td>NUMBER</td>
<td>No</td>
<td>The planned work quantity for the task</td>
</tr>
<tr>
<td>P_PLANNED_EFFORT</td>
<td>NUMBER</td>
<td>No</td>
<td>The planned effort for the task</td>
</tr>
</tbody>
</table>

**TASK_OUT_TBL_TYPE Datatype**

The following table shows the TASK_OUT_TBL_TYPE datatype.
### Name, Type, Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ID</td>
<td>NUMBER (15)</td>
<td>The code that uniquely identifies a task in a project in Oracle Projects</td>
</tr>
<tr>
<td>PM_TASK_REFERENCE</td>
<td>VARCHAR2 (25)</td>
<td>The reference code that identifies a project's task in the external system</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>VARCHAR2 (1)</td>
<td>API standard</td>
</tr>
<tr>
<td>TASK_VERSION_ID</td>
<td>NUMBER</td>
<td>Task Version ID</td>
</tr>
</tbody>
</table>

**TASK_IN_REC_TYPE Datatype**

The following table shows the TASK_IN_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETIREMENT_COST_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that identifies tasks for retirement cost collection</td>
</tr>
<tr>
<td>CINT_ELIGIBLE_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether the project is eligible for capitalized interest</td>
</tr>
<tr>
<td>CINT_STOP_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Stop date for capital interest calculation</td>
</tr>
<tr>
<td>PRED_STRING</td>
<td>VARCHAR2 (4000)</td>
<td>Yes</td>
<td>The string containing the predecessors information</td>
</tr>
<tr>
<td>PRED_DELIMITER</td>
<td>VARCHAR2 (1)</td>
<td>Yes</td>
<td>Delimiter that separates predecessors in the predecessor string</td>
</tr>
<tr>
<td>BASE_PERCENT_COMP_DERIV_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>Base percent complete derivation code for the task</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SCH_TOOL_TSK_TYPE_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>Default scheduling tool task type for the task version</td>
</tr>
<tr>
<td>CONSTRAINT_TYPE_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>Constraint type for the task version</td>
</tr>
<tr>
<td>CONSTRAINT_DATE</td>
<td>DATE</td>
<td>Yes</td>
<td>Constraint date for the task version</td>
</tr>
<tr>
<td>FREE_SLACK</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Free slack for the task version</td>
</tr>
<tr>
<td>TOTAL_SLACK</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Total slack for the task version</td>
</tr>
<tr>
<td>EFFORT_DRIVEN_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>Yes</td>
<td>The flag that indicates whether the task is effort driven</td>
</tr>
<tr>
<td>LEVEL_ASSIGNMENTS_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>Yes</td>
<td>The flag that indicates whether the assignments on this task should be leveled</td>
</tr>
<tr>
<td>INVOICE_METHOD</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>The invoice method for the task. Valid only for top tasks with Invoice Method enabled.</td>
</tr>
<tr>
<td>CUSTOMER_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The customer for the task. Valid only for top tasks with Customer enabled.</td>
</tr>
<tr>
<td>GEN_ETC_SOURCE_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>Estimate to complete source</td>
</tr>
<tr>
<td>FINANCIAL_TASK_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>Yes</td>
<td>The flag that indicates whether the task is a financial task or not. This flag is valid only for partially shared structures. Tasks that are above this level are used for financial management.</td>
</tr>
<tr>
<td>MAPPED_TASK_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Mapped task ID</td>
</tr>
<tr>
<td>MAPPED_TASK_REFERENCE</td>
<td>VARCHAR2 (150)</td>
<td>Yes</td>
<td>Mapped task reference</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>DELIVERABLE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Deliverable reference to be associated with the task</td>
</tr>
<tr>
<td></td>
<td>(4000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELIVERABLE_ID</td>
<td>NUMBER</td>
<td></td>
<td>Identification code of the deliverable associated with the task</td>
</tr>
<tr>
<td>EXT_ACT_DURATION</td>
<td>NUMBER</td>
<td>Yes</td>
<td>From the external application, the actual duration</td>
</tr>
<tr>
<td>EXT_REMAIN_DUR</td>
<td>NUMBER</td>
<td>Yes</td>
<td>From the external application, the remaining duration</td>
</tr>
<tr>
<td>EXTERNAL</td>
<td>NUMBER</td>
<td>Yes</td>
<td>From the external application, the scheduled duration</td>
</tr>
<tr>
<td>ETC_EFFORT</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Estimated remaining effort for the task</td>
</tr>
<tr>
<td>PERCENT_COMPLETE</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Percentage of work complete on the task</td>
</tr>
</tbody>
</table>

**CUSTOMER_TBL_TYPE Datatype**

This record type enables the user to pass multiple customers to the UPDATE_PROJECT API. The following table shows the CUSTOMER_TBL_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUSTOMER_ID</td>
<td>NUMBER</td>
<td></td>
<td>Based on template setup The identifier of the project customer</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_RELATIONSHIP_CODE</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>The identifier of the customer relationship type that the customer has with the project</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BILL_TO_CUSTOMER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the customer to whom invoices are sent</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIP_TO_CUSTOMER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the customer that is the project customer's default work site for the project</td>
</tr>
<tr>
<td></td>
<td>(15)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BILL_TO_ADDRESS_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the customer address to which invoices are sent</td>
</tr>
<tr>
<td>SHIP_TO_ADDRESS_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the customer address that is the project customer's default work site for the project</td>
</tr>
<tr>
<td>CONTACT_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The identifier of the contact who represents the customer for the project</td>
</tr>
<tr>
<td>PROJECT_CONTACT_TYPE_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The contact type that classifies the contact</td>
</tr>
<tr>
<td>CUSTOMER_BILL_SPLIT</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The percentage of the total project revenue and invoice that the customer is charged</td>
</tr>
<tr>
<td>ALLOW_INV_USER_RATE_TYPE_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag indicating whether the User exchange rate type is permitted for the customer on the project.</td>
</tr>
<tr>
<td>INV_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The default invoice currency exchange rate date for draft invoices generated for this project customer</td>
</tr>
<tr>
<td>INV_RATE_TYPE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Default invoice currency exchange rate type for draft invoices generated for this project customer</td>
</tr>
<tr>
<td>INV_CURRENCY_CODE</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>Default invoice currency code for draft invoices generated for this project customer</td>
</tr>
<tr>
<td>INV_EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Default invoice currency exchange rate for draft invoices generated for this project customer</td>
</tr>
<tr>
<td>BILL_ANOTHER_PROJECT_FLAG</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Flag indicating whether the project customer is internal</td>
</tr>
</tbody>
</table>
### Oracle Project Foundation APIs

**RECEIVER_TASK_ID**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECEIVER_TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Task identifier of the receiver task that is linked to the internal project customer</td>
</tr>
</tbody>
</table>

**DELIVERABLE_IN_REC_TYPE**

The following table shows the DELIVERABLE_IN_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELIVERABLE_SHORT_NAME</td>
<td>VARCHAR2 2000</td>
<td>Yes</td>
<td>Short name of the deliverable</td>
</tr>
<tr>
<td>DELIVERABLE_NAME</td>
<td>VARCHAR2 240</td>
<td>Yes</td>
<td>Long name of the deliverable</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2 2000</td>
<td>No</td>
<td>Description of the deliverable</td>
</tr>
<tr>
<td>DELIVERABLE_OWNER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>ID of the deliverable owner</td>
</tr>
<tr>
<td>STATUS_CODE</td>
<td>VARCHAR2 30</td>
<td>No</td>
<td>Status of the deliverable</td>
</tr>
<tr>
<td>DELIVERABLE_TYPE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>ID of the deliverable type</td>
</tr>
<tr>
<td>PROGRESS_WEIGHT</td>
<td>NUMBER</td>
<td>No</td>
<td>Progress weight of the deliverable</td>
</tr>
<tr>
<td>DUE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Deliverable due date</td>
</tr>
<tr>
<td>COMPLETION_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Deliverable completion date</td>
</tr>
<tr>
<td>PM_SOURCE_CODE</td>
<td>VARCHAR2 30</td>
<td>No</td>
<td>Identifier of the external system</td>
</tr>
<tr>
<td>PM_DELIVERABLE_REFERENCE</td>
<td>VARCHAR2 25</td>
<td>Yes</td>
<td>The unique identifier of the deliverable in the external system</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DELIVERABLE_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The unique identifier of the deliverable in Oracle Projects</td>
</tr>
<tr>
<td>TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The task ID of the task where the deliverable is created</td>
</tr>
<tr>
<td>TASK_SOURCE_REFERENCE</td>
<td>VARCHAR2(25)</td>
<td>No</td>
<td>The task reference of the task where the deliverable is created</td>
</tr>
<tr>
<td>ITEM_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The ID of the item. This ITEM_ID applies only to item deliverables.</td>
</tr>
<tr>
<td>INVENTORY_ORG_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The inventory org ID of the item. INVENTORY_ORG_ID applies only to item deliverables.</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>NUMBER</td>
<td>No</td>
<td>The quantity of the item. QUANTITY applies only to item deliverables.</td>
</tr>
<tr>
<td>UOM_CODE</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The unit of measure code of the deliverable. UOM_CODE applies only to item deliverables.</td>
</tr>
<tr>
<td>UNIT_PRICE</td>
<td>NUMBER</td>
<td>No</td>
<td>The unit price of the deliverable. UNIT_PRICE applies only to item deliverables.</td>
</tr>
<tr>
<td>UNIT_NUMBER</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The unit number of the deliverable. UNIT_NUMBER is needed when an item deliverable is unit number enabled.</td>
</tr>
<tr>
<td>CURRENCY_CODE</td>
<td>VARCHAR2(15)</td>
<td>No</td>
<td>The currency code of the deliverable. CURRENCY_CODE applies only to item deliverables.</td>
</tr>
</tbody>
</table>

**DELIVERABLE_OUT_REC_TYPE**

The following table shows the DELIVERABLE_OUT_REC_TYPE datatype.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELIVERABLE_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The unique identifier of the deliverable in Oracle Projects</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>VARCHAR2</td>
<td></td>
<td>API standard</td>
</tr>
</tbody>
</table>

**ACTION_IN_REC_TYPE**

The following table shows the ACTION_IN_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION_NAME</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>Name of the deliverable action</td>
</tr>
<tr>
<td></td>
<td>(100)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTION_OWNER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>ID of the deliverable action owner</td>
</tr>
<tr>
<td>ACTION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The unique identifier of the deliverable action in Oracle Projects</td>
</tr>
<tr>
<td>FUNCTION_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The deliverable action function code</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The due date of the deliverable action</td>
</tr>
<tr>
<td>COMPLETION_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The completion date of the deliverable action</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Description of the deliverable action</td>
</tr>
<tr>
<td></td>
<td>(2000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_SOURCE_CODE</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Identifier of the external system</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_ACTION_REFERENCE</td>
<td>VARCHAR2</td>
<td>Yes</td>
<td>The unique identifier of the deliverable action in the external system</td>
</tr>
<tr>
<td></td>
<td>(25)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PM_DELIVERABLE_REFERENCE</td>
<td>VARCHAR2 (25)</td>
<td>Yes</td>
<td>The unique identifier of the deliverable in the external system</td>
</tr>
<tr>
<td>DELIVERABLE_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The unique identifier of the deliverable in Oracle Projects</td>
</tr>
<tr>
<td>FINANCIAL_TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Financial task ID</td>
</tr>
<tr>
<td>FINANCIAL_TASK_REFERENCE</td>
<td>VARCHAR2 (25)</td>
<td>No</td>
<td>The unique identifier of the financial task in the external system</td>
</tr>
<tr>
<td>DESTINATION_TYPE_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The destination type code of the deliverable action</td>
</tr>
<tr>
<td>RECEIVING_ORG_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The inventory org ID of the inventory organization that will receive the item</td>
</tr>
<tr>
<td>RECEIVING_LOCATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The location (address) where the item will be received. The RECEIVING_LOCATION_ID must be related to the RECEIVING_ORG_ID.</td>
</tr>
<tr>
<td>PO_NEED_BY_DATE</td>
<td>DATE</td>
<td>No</td>
<td>The date by which the purchase order is needed</td>
</tr>
<tr>
<td>VENDOR_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The ID of the vendor that will supply the object</td>
</tr>
<tr>
<td>VENDOR_SITE_CODE</td>
<td>VARCHAR2 (15)</td>
<td>No</td>
<td>The location (address) code of the vendor from where the object will be supplied</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>NUMBER</td>
<td>No</td>
<td>Quantity for procurement deliverable action</td>
</tr>
<tr>
<td>UOM_CODE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The unit of measure code for procurement deliverable action</td>
</tr>
<tr>
<td>UNIT_PRICE</td>
<td>NUMBER</td>
<td>No</td>
<td>The unit price of the object. The unit price is needed only for non-item deliverable procurement actions.</td>
</tr>
<tr>
<td>EXCHANGE_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Exchange rate type for procurement deliverable actions</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EXCHANGE_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Exchange rate date for procurement deliverable actions</td>
</tr>
<tr>
<td>EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Exchange rate for procurement deliverable actions</td>
</tr>
<tr>
<td>EXPENDITURE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>The expenditure type is needed only for non-item deliverable procurement actions.</td>
</tr>
<tr>
<td>EXPENDITURE_ORG_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Expenditure organization for procurement deliverable actions</td>
</tr>
<tr>
<td>EXPENDITURE_ITEM_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Expenditure item date for procurement deliverable actions</td>
</tr>
<tr>
<td>REQUISITION_LINE_TYPE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The requisition line type can only have a type of &quot;AMOUNT&quot;. The requisition line type is needed only for non-item deliverable procurement actions.</td>
</tr>
<tr>
<td>CATEGORY_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The item category ID is needed only for non-item deliverable procurement actions.</td>
</tr>
<tr>
<td>READY_TOPROCURE_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether an item is ready to procure</td>
</tr>
<tr>
<td>INITIATEPROCURE_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether a procurement action should be initiated</td>
</tr>
<tr>
<td>SHIP_FROM_ORGANIZATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The inventory org ID of the inventory organization that will ship the item</td>
</tr>
<tr>
<td>SHIP_FROM_LOCATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The location (address) where the item will be shipped. The SHIP_FROM_LOCATION_ID must be related to the SHIP_FROM_ORG_ID.</td>
</tr>
<tr>
<td>SHIP_TO_ORGANIZATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The customer account ID to which the item will be shipped</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SHIP_TO_LOCATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>The customer location (address) to where the item will be shipped. The SHIP_TO_LOCATION_ID must be related to the SHIP_TO_ORG_ID.</td>
</tr>
<tr>
<td>DEMAND_SCHEDULE</td>
<td>VARCHAR2 (10)</td>
<td>No</td>
<td>Demand schedule</td>
</tr>
<tr>
<td>EXPECTED_SHIPMENT_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Expected shipment date</td>
</tr>
<tr>
<td>PROMISED_SHIPMENT_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Promised shipment date</td>
</tr>
<tr>
<td>VOLUME</td>
<td>NUMBER</td>
<td>No</td>
<td>The volume of each object to be shipped. The volume is needed only for non-item shipping.</td>
</tr>
<tr>
<td>VOLUME_UOM</td>
<td>VARCHAR2 (10)</td>
<td>No</td>
<td>The volume unit of measure of each object to be shipped. The volume unit of measure is needed only for non-item shipping.</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>NUMBER</td>
<td>No</td>
<td>The weight of each object to be shipped. The weight is needed only for non-item shipping.</td>
</tr>
<tr>
<td>WEIGHT_UOM</td>
<td>VARCHAR2 (10)</td>
<td>No</td>
<td>The weight unit of measure of each object to be shipped. The weight unit of measure is needed only for non-item shipping.</td>
</tr>
<tr>
<td>READY_TO_SHIP_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether an item is ready to ship</td>
</tr>
<tr>
<td>INITIATE_PLANNING_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether planning should be initiated</td>
</tr>
<tr>
<td>INITIATE_SHIPPING_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether shipping should be initiated</td>
</tr>
<tr>
<td>EVENT_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Billing event type</td>
</tr>
</tbody>
</table>

---

3-52  Oracle Projects APIs, Client Extensions, and Open Interfaces Reference
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENCY</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Currency code for the billing event</td>
</tr>
<tr>
<td>INVOICE_AMOUNT</td>
<td>NUMBER</td>
<td>No</td>
<td>Invoice amount for the billing event</td>
</tr>
<tr>
<td>REVENUE_AMOUNT</td>
<td>NUMBER</td>
<td>No</td>
<td>Revenue amount for the billing event</td>
</tr>
<tr>
<td>EVENT_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Event date for the billing event</td>
</tr>
<tr>
<td>EVENT_NUMBER</td>
<td>NUMBER</td>
<td>No</td>
<td>Event number for the billing event</td>
</tr>
<tr>
<td>ORGANIZATION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Organization ID of the organization associated with the billing event</td>
</tr>
<tr>
<td>BILL_HOLD_FLAG</td>
<td>VARCHAR2 (1)</td>
<td>No</td>
<td>Flag that indicates whether a billing event is on hold</td>
</tr>
<tr>
<td>PROJECT_FUNCTIONAL_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Rate type for project functional currency for the billing event</td>
</tr>
<tr>
<td>PROJECT_FUNCTIONAL_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Rate date for project functional currency for the billing event</td>
</tr>
<tr>
<td>PROJECT_FUNCTIONAL_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Rate for project functional currency for the billing event</td>
</tr>
<tr>
<td>PROJECT_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Rate type for project currency for the billing event</td>
</tr>
<tr>
<td>PROJECT_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Rate date for project currency for the billing event</td>
</tr>
<tr>
<td>PROJECT_RATE</td>
<td>NUMBER</td>
<td>No</td>
<td>Rate for project currency for the billing event</td>
</tr>
<tr>
<td>FUNDING_RATE_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>No</td>
<td>Rate type for funding currency for the billing event</td>
</tr>
<tr>
<td>FUNDING_RATE_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Rate date for funding currency for the billing event</td>
</tr>
</tbody>
</table>
Using Project Definition APIs

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 2-21.
- 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 3-60.

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 2-8.
**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:
**Example of a Quick Entry (Overridable Fields) Window**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Mandatory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name</td>
<td>Build Castle</td>
<td>✗</td>
</tr>
<tr>
<td>Distribution Rule</td>
<td>COST/COST</td>
<td>✓</td>
</tr>
<tr>
<td>Project Manager</td>
<td>t</td>
<td>✓</td>
</tr>
<tr>
<td>Funding Source</td>
<td>Federal</td>
<td>✓</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Quick Entry Field Value</th>
<th>LOAD_PROJECT PARAMETER</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>P_PROJECT_NAME</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>P_DESCRIPTION</td>
</tr>
<tr>
<td>START_DATE</td>
<td>P_START_DATE</td>
</tr>
<tr>
<td>COMPLETION_DATE</td>
<td>P_COMPLETION_DATE</td>
</tr>
<tr>
<td>PROJECT_STATUS_CODE</td>
<td>P_PROJECT_STATUS_CODE</td>
</tr>
<tr>
<td>PUBLIC_SECTOR_FLAG</td>
<td>P_PUBLIC_SECTOR_FLAG</td>
</tr>
<tr>
<td>DISTRIBUTION_RULE</td>
<td>P_DISTRIBUTION_RULE</td>
</tr>
<tr>
<td>CARRYING_OUT_ORGANIZATION_ID</td>
<td>P_CARRYING_OUT_ORGANIZATION_ID</td>
</tr>
<tr>
<td>CUSTOMER_NAME</td>
<td>P_CUSTOMER_ID</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Key Member Quick Entry Field</th>
<th>Input Parameter for LOAD_KEY_MEMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_MEMBER (value)</td>
<td>P_PERSON_ID</td>
</tr>
<tr>
<td>KEY_MEMBER (display_name)</td>
<td>P_PROJECT_ROLE_TYPE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Class Category Quick Entry Field</th>
<th>Input Parameter for LOAD_CLASSCATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLASS_CATEGORY (value)</td>
<td>P_CLASS_CODE</td>
</tr>
<tr>
<td>CLASSCATEGORY (display_name)</td>
<td>P_CLASSCATEGORY</td>
</tr>
</tbody>
</table>

**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.

**Important:** 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
Step 7: Start the Server-Side Process

After 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 2-7.

- **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 2-25.

**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.

```sql
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_type;
l_key_member_tbl pa_project_pub.project_role_tbl_type;
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;
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 multiple 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_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 ;
/

**Structure APIs**

The structure APIs enable you to use an external system to create and change structure versions. The structure APIs also enable you to add, update, and delete tasks.
**Note:** When you call any structure 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.

### Structure API Views

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

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_PROJECT_STATUS_LOV_V</td>
<td>Retrieves project statuses from Oracle Projects</td>
</tr>
<tr>
<td>PA_SERVICE_TYPE_LOV_V</td>
<td>Because valid service type codes must be selected for the parameter service_type_code, you can use this view to retrieve valid codes for service_type_code from Oracle Projects and display them in your external system.</td>
</tr>
<tr>
<td>PA_TASK_MANAGERS_LOV_V</td>
<td>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.</td>
</tr>
<tr>
<td>PA_TASKS_AMG_V</td>
<td>Retrieves information about all valid tasks for the organization associated with the user’s responsibility. This view provides a list of financial tasks and their related attributes.</td>
</tr>
<tr>
<td>PA_TASKS_LOWEST_V</td>
<td>A supplementary view used to simplify coding in forms.</td>
</tr>
<tr>
<td>PA_TASK_PROGRESS_AMG_V</td>
<td>Retrieves information about all valid task progress for the organization associated with the user’s responsibility.</td>
</tr>
<tr>
<td>PA_STRUCT_TASKS_AMG_V</td>
<td>You can use this view to retrieve valid structures from Oracle Projects and display them in your external system. This list provides a list of tasks and their related attributes.</td>
</tr>
</tbody>
</table>
View | Description
---|---
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. This view provides a list of structure versions.
PA_TASK_INV_METHODS_LOV_V | This view retrieves the invoice methods for the tasks.
PA_TOP_TASK_CUSTOMERS_LOV_V | This view retrieves all of the customers for a project.

**Structure API Procedures**

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

- Structure and Task Procedures
  - ADD_TASK, page 3-69
  - APPLY_LP_PROG_ON_CWV, page 3-72
  - BASELINE_STRUCTURE, page 3-72
  - CHANGE_CURRENT_WORKING_VERSION, page 3-73
  - CHANGE_STRUCTURE_STATUS, page 3-73
  - DELETE_STRUCTURE_VERSION, page 3-74
  - DELETE_TASK, page 3-74
  - GET_TASK_VERSION, page 3-75
  - GET_DELETED_TASKS_FROM_OP, page 3-76
  - UPDATE_TASK, page 3-76

- Load-Execute-Fetch Procedures
  - FETCH_STRUCTURE_VERSION, page 3-80
  - FETCH_TASK, page 3-80
• FETCH_TASKS, page 3-80
• FETCH_TASK_VERSION, page 3-81
• LOAD_STRUCTURE, page 3-81
• LOAD_TASK, page 3-81
• LOAD_TASKS, page 3-83

• Check Procedures
  • CHECK_ADD_SUBTASK_OK, page 3-83
  • CHECK_CHANGE_PARENT_OK, page 3-83
  • CHECK_DELETE_TASK_OK, page 3-83
  • CHECK_TASK_NUMBER_CHANGE_OK, page 3-84
  • CHECK_TASK_MFD, page 3-84
  • CHECK_UNIQUE_TASK_NUMBER, page 3-84
  • CHECK_UNIQUE_TASK_REFERENCE, page 3-85

Structure APIs Procedure Definitions
This section contains description of the structure APIs, including business rules and parameters.

ADD_TASK
ADD_TASK is a PL/SQL procedure used to add new subtasks to a task of a project in Oracle Projects. The task record type has been replaced with a parameter with a standard datatype (NUMBER, VARCHAR2, or DATE) for every field in the record type definition so that 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.
Performing Scheduling Validations

When set to 'N' the P_OP_VALIDATE_FLAG parameter eliminates redundant validation for certain types of scheduling data. Unnecessary scheduling validations can slow system performance.

You should set P_OP_VALIDATE_FLAG to 'N' if you are using this API to integrate a third-party scheduling tool with Oracle Projects. Only set this parameter to 'N' if the third-party scheduling tool can perform validations for:

- Dependencies between tasks and activities
- Project schedule dates and task schedule dates

You should set P_OP_VALIDATE_FLAG to 'Y' if you are using the APIs to upload data from a legacy system that does not perform an extensive validation for the above. Scheduling data must be validated in order to ensure data integrity in Oracle Projects.

Parameters for ADD_TASK

You can view descriptions of all of the parameters for ADD_TASK in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for ADD_TASK are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
- P_STRUCTURE_VERSION_ID
- P_PA_PROJECT_ID
- P_PM_TASK_REFERENCE
- P_PA_TASK_NUMBER
- P_TASK_NAME
- P_PRED_STRING
- P_PRED_DELIMITER
- P_BASE_PERCENT_COMP_DERIV_CODE
- P_SCH_TOOL_TSK_TYPE_CODE
• P_CONSTRAINT_TYPE_CODE
• P_CONSTRAINT_DATE
• P_FREE_SLACK
• P_TOTAL_SLACK
• P_EFFORT_DRIVEN_FLAG
• P_LEVEL_ASSIGNMENTS_FLAG
• P_INVOICE_METHOD
• P_CUSTOMER_ID
• P_GEN_ETC_SOURCE_CODE
• P_MAPPED_TASK_ID
• P_MAPPED_TASK_REFERENCE
• P_EXT_ACT_DURATION
• P_EXT_REMAIN_DURATION
• P_EXT_SCH_DURATION

**APPLY_LP_PROG_ON_CWV**

APPLY_LP_PROG_ON_CWV is used to apply the latest progress information on the current working version of a structure.

You can view descriptions of all of the parameters for APPLY_LP_PROG_ON_CWV in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for APPLY_LP_PROG_ON_CWV are listed below:

• P_API_VERSION_NUMBER
• P_PM_WORKING_STR_VERSION_ID

**BASELINE_STRUCTURE**

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

You can view descriptions of all of the parameters for BASELINE_STRUCTURE in the Oracle Integration Repository. The Oracle Integration Repository is described in the
The required parameters for BASELINE_STRUCTURE are listed below:

- P_API_VERSION_NUMBER
- P_STRUCTURE_VERSION_ID
- P_PA_PROJECT_ID

**CHANGE_CURRENT_WORKING_VERSION**

CHANGE_CURRENT_WORKING_VERSION is used to change the current working version of a structure.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_STRUCTURE_VERSION_ID
- P_PA_PROJECT_ID

**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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
• P_STRUCTURE_VERSION_ID
• P_PA_PROJECT_ID
• P_STATUS_CODE

DELETE_STRUCTURE_VERSION

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_STRUCTURE_VERSION_ID

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:
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):

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

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

GET_TASK_VERSION

GET_TASK_VERSION is used to get the task version ID of a task for a particular structure version.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PA_PROJECT_ID
- P_PA_TASK_ID
- P_PA_STRUCTURE_VERSION_ID
GET_DELETED_TASKS_FROM_OP

When you publish a version-enabled workplan in Oracle Projects, tasks in the working version with a status of To Be Deleted are either deleted or set to a Cancelled status. The GET_DELETED_TASKS_FROM_OP procedure retrieves the list of deleted tasks in Oracle Projects and displays the tasks in an integrated external system such as Microsoft Project.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

All of the parameters for this procedure are OUT parameters; therefore, there are no required parameters.

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

Performing Scheduling Validations

When set to 'N' the P_OP_VALIDATE_FLAG parameter eliminates redundant validation for certain types of scheduling data. Unnecessary scheduling validations can slow system performance.

You should set P_OP_VALIDATE_FLAG to 'N' if you are using this API to integrate a third-party scheduling tool with Oracle Projects. Only set this parameter to 'N' if the third-party scheduling tool can perform validations for:

- Dependencies between tasks and activities
- Project schedule dates and task schedule dates

You should set P_OP_VALIDATE_FLAG to 'Y' if you are using the APIs to upload data from a legacy system that does not perform an extensive validation for the above. Scheduling data must be validated in order to ensure data integrity in Oracle Projects.

Parameters for UPDATE_TASK

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of
The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
- P_PA_PROJECT_ID
- P_PM_TASK_REFERENCE
- P_PA_TASK_ID
- P_TASK_NAME
- P_PA_TASK_NUMBER
- P_TASK_DESCRIPTION
- P_PRED_STRING
- P_PRED_DELIMITER
- P_BASE_PERCENT_COMP_DERIV_CODE
- P_SCH_TOOL_TSK_TYPE_CODE
- P_CONSTRAINT_TYPE_CODE
- P_CONSTRAINT_DATE
- P_FREE_SLACK
- P_TOTAL_SLACK
- P_EFFORT_DRIVEN_FLAG
- P_LEVEL_ASSIGNMENTS_FLAG
- P_INVOICE_METHOD
- P_CUSTOMER_ID
- P_GEN_ETC_SOURCE_CODE
- P_FINANCIAL_TASK_FLAG
• P_MAPPED_TASK_ID
• P_MAPPED_TASK_REFERENCE
• P_DELIVERABLE
• P_EXT_ACT_DURATION
• P_EXT_REMAIN_DURATION
• P_EXT_SCH_DURATION
• P_ETC_EFFORT
• P_PERCENT_COMPLETE

FETCH_STRUCTURE_VERSION

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER

FETCH_TASK

FETCH_TASK is a Load-Execute-Fetch procedure used to fetch output parameters related to tasks. FETCH_TASK is used to load successfully processed tasks to a global PL/SQL table.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_TASK_INDEX

FETCH_TASKS

FETCH_TASKS is a wrapper for FETCH_TASK to handle multiple calls to FETCH_TASK.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- \texttt{P\_API\_VERSION\_NUMBER}
- \texttt{P\_TASK\_INDEX}

\textbf{FETCH\_TASK\_VERSION}

FETCH\_TASK\_VERSION is a Load-Execute-Fetch procedure that returns version identifiers of tasks.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- \texttt{P\_API\_VERSION\_NUMBER}
- \texttt{P\_TASK\_INDEX}

\textbf{LOAD\_STRUCTURE}

LOAD\_STRUCTURE is a Load-Execute-Fetch procedure used to load structure data.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- \texttt{P\_API\_VERSION\_NUMBER}

\textbf{LOAD\_TASK}

LOAD\_TASK is a Load-Execute-Fetch procedure used to load a task to a global PL/SQL table.

\textbf{Business Rule (task level)}

Oracle Projects imposes the following business rule:

- Parent tasks must be loaded before their subtasks.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of
The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_TASK_NAME
- P_PA_TASK_NUMBER
- P_PRED_STRING
- P_PRED_DELIMITER
- P_BASE_PERCENT_COMP_DERIV_CODE
- P_SCH_TOOL_TSK_TYPE_CODE
- P_CONSTRAINT_TYPE_CODE
- P_CONSTRAINT_DATE
- P_FREE_SLACK
- P_TOTAL_SLACK
- P_EFFORT_DRIVEN_FLAG
- P_LEVEL_ASSIGNMENTS_FLAG
- P_INVOICE_METHOD
- P_CUSTOMER_ID
- P_GEN_ETC_SOURCE_CODE
- P_FINANCIAL_TASK_FLAG
- P_MAPPED_TASK_ID
- P_MAPPED_TASK_REFERENCE
- P_EXT_ACT_DURATION
- P_EXT_REMAIN_DURATION
- P_EXT_SCH_DURATION
- P_ETC_EFFORT
• P_PERCENT_COMPLETE

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 those for LOAD_TASK, page 3-81.

Check Procedures
   The following check procedures are PL/SQL procedures used to verify in real time that:
   • Task information you have entered into your external system is unique in Oracle Projects
   • Certain functions, such as deleting a 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.

   You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

   The required parameters for this procedure are listed below:
   • P_API_VERSION_NUMBER

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.

   You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

   The required parameters for this procedure are listed below:
   • P_API_VERSION_NUMBER

CHECK_DELETE_TASK_OK
   Use the Check procedure CHECK_DELETE_TASK_OK to determine if you can delete a
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER

**CHECK_TASK_NUMBER_CHANGE_OK**

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER

**CHECK_TASK_MFD**

Use the check procedure CHECK_TASK_MFD to determine whether tasks deleted in an external scheduling system such as Microsoft Project can be deleted in Oracle Projects when that system is integrated with Oracle Projects. Tasks are deleted immediately in Oracle Projects if no published version exists, and are marked for deletion or cancelled when the working workplan version is published in Oracle Projects. This procedure prevents tasks from being deleted if they have transactions.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER

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 3-17
• Execute_Update_Project, page 3-18

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

LOAD_EXTENSIBLE_ATTRIBUTE

This API loads a single attribute value for a given attribute group for the specified project and task.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The following table shows which parameters for this procedure are required:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_API_VERSION_NUMBER</td>
<td>Yes</td>
</tr>
<tr>
<td>P_TASK_ID</td>
<td>1 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_TASK_REFERENCE</td>
<td>1 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_GRP_INTERNAL_NAME</td>
<td>2 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_GRP_ID</td>
<td>2 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_GRP_ROW_INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>P_ATTR_VALUE_STR</td>
<td>3 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_VALUE_NUM</td>
<td>3 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_VALUE_DATE</td>
<td>3 (See Parameter Requirements, page 3-87)</td>
</tr>
</tbody>
</table>
Parameter Requirements

In the preceding 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_TASKREFERENCE 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The following table shows the required parameters for this procedure:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_TASK_ID</td>
<td>1 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_TASK_REFERENCE</td>
<td>1 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_GRP_INTERNAL_NAME</td>
<td>2 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_GRP_ID</td>
<td>2 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_GRP_ROW_INDEX</td>
<td>Yes</td>
</tr>
<tr>
<td>P_ATTR_VALUE_STR</td>
<td>3 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_VALUE_NUM</td>
<td>3 (See Parameter Requirements, page 3-87)</td>
</tr>
<tr>
<td>P_ATTR_VALUE_DATE</td>
<td>3 (See Parameter Requirements, page 3-87)</td>
</tr>
</tbody>
</table>

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.

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.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM Project Type</td>
<td>AIM</td>
</tr>
<tr>
<td>Process Change Required</td>
<td>Low</td>
</tr>
</tbody>
</table>
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 are identified by using a common Attribute Row Identifier.

**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.

<table>
<thead>
<tr>
<th>PL/SQL Record Number</th>
<th>Attribute Row Identifier</th>
<th>Internal Attribute Group Name</th>
<th>Internal Attribute Name</th>
<th>Attribute Value (String)</th>
<th>Attribute Value (Number)</th>
<th>Attribute Value (Date)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Project Complexity</td>
<td>AIM Project Type</td>
<td>[blank]</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Project Complexity</td>
<td>Process Change Required</td>
<td>Low</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Project Complexity</td>
<td>System Size/Complexity</td>
<td>Medium</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Project Complexity</td>
<td>Customization Required</td>
<td>Medium</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Project Complexity</td>
<td>Implementation Type</td>
<td>Phased</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Project Complexity</td>
<td>Complexity Score</td>
<td>[blank]</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
</tbody>
</table>
Integrating Multi-Row Attribute Groups

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

<table>
<thead>
<tr>
<th>Product Family</th>
<th>Application Module</th>
<th>Default Weightings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financials</td>
<td>Assets</td>
<td>0.4</td>
</tr>
<tr>
<td>Financials</td>
<td>General Ledger</td>
<td>0.6</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>PL/SQL Record Number</th>
<th>Attribute Row Identifier</th>
<th>Internal Attribute Group Name</th>
<th>Internal Attribute Name</th>
<th>Attribute Value (String)</th>
<th>Attribute Value (Number)</th>
<th>Attribute Value (Date)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Application Weightings</td>
<td>Product Family</td>
<td>[blank]</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Application Weightings</td>
<td>Application Module</td>
<td>[blank]</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Application Weightings</td>
<td>Default Weighting</td>
<td>[blank]</td>
<td>0.4</td>
<td>[blank]</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>Application Weightings</td>
<td>Product Family</td>
<td>[blank]</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Application Weightings</td>
<td>Application Module</td>
<td>[blank]</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>Application Weightings</td>
<td>Default Weighting</td>
<td>[blank]</td>
<td>0.6</td>
<td>[blank]</td>
</tr>
</tbody>
</table>
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_tasklevels 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-00';
l_actual_finish_date := '01-jan-00';
l_early_start_date := '01-jan-00';
l_early_finish_date := '01-jan-00';
l_late_start_date := '01-jan-00';
l_late_finish_date := '01-jan-00';
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';
end loop;
END;
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_final_msg );
  END LOOP;
END IF;
-----------------------
--dbms_output.put_line('bef execute create project');
-----------------------
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');

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

--CLEAR_CREATE_PROJECT
pa_project_pub.clear_project;
IF l_return_status != 'S' THEN
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 project 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_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 NUMBER;
3-100 Oracle Projects APIs, Client Extensions, and Open Interfaces Reference
--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;
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;
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.planned_task_reference := '10-MAR-95';
  l_task_rec.actual_start_date := '09-MAR-95';
  l_task_rec.actual_finish_date := '05-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;
  end loop;
  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;
  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;
  end loop;
  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;
  end loop;
end loop;
:= 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_attr_value_num.extend;
l_attr_value_date.extend;
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_attr_value_num.extend;
l_attr_value_date.extend;
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_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;
-----------------------
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.
<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_AMGRESOURCE_INFO_V</td>
<td>Customize this view to retrieve information about resource list members.</td>
</tr>
<tr>
<td>PA_EMPLOYEES_RES_V</td>
<td>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.</td>
</tr>
<tr>
<td>PA_EVENT_TYPES_RES_V</td>
<td>Displays event types defined in Oracle Projects. You can define any event type returned by this view as a resource in Oracle Projects.</td>
</tr>
<tr>
<td>PA_EXPEND_CATEGORIES_RES_V</td>
<td>Displays expenditure categories defined in Oracle Projects. You can define any expenditure category returned by this view as a resource in Oracle Projects.</td>
</tr>
<tr>
<td>PA_EXPENDITURE_TYPES_RES_V</td>
<td>Displays expenditure types defined in Oracle Projects. You can define any expenditure type returned by this view as a resource in Oracle Projects.</td>
</tr>
<tr>
<td>PA_JOBS_RES_V</td>
<td>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.</td>
</tr>
<tr>
<td>PA_LOWEST_LEVEL_RESOURCES_V</td>
<td>Retrieves Oracle Projects identification codes and names for resource lists and lowest-level resource list members.</td>
</tr>
<tr>
<td>PA_ORGANIZATIONS_RES_V</td>
<td>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.</td>
</tr>
<tr>
<td>PA_PROJ_ORG_STRUCTURES_V</td>
<td>Retrieves the organization hierarchy</td>
</tr>
<tr>
<td>PA_QRYRESOURCE_LISTS_V</td>
<td>Retrieves resource lists defined in Oracle Projects</td>
</tr>
<tr>
<td>PA_QUERY_RES_LIST_MEMBERS_V</td>
<td>Retrieves members of a resource list defined in Oracle Projects</td>
</tr>
<tr>
<td>PA_RESOURCE_LISTGROUPS_V</td>
<td>Retrieves resource groups in a resource list defined in Oracle Projects</td>
</tr>
<tr>
<td>PA_RESOURCE_LIST_V</td>
<td>Retrieves resource lists defined in Oracle Projects</td>
</tr>
<tr>
<td>PA_RESOURCETYPES_ACTIVE_V</td>
<td>Retrieves active resource types defined in Oracle Projects</td>
</tr>
<tr>
<td>View</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_REVENUE_CATEGORIES_RES_V</td>
<td>Displays revenue categories defined in Oracle Projects. You can define any revenue category returned by this view as a resource in Oracle Projects.</td>
</tr>
<tr>
<td>PA_VENDORS_RES_V</td>
<td>Displays information about vendors defined in Oracle Purchasing. You can define any vendor returned by this view as a resource in Oracle Projects.</td>
</tr>
</tbody>
</table>

**Resource API Procedures**

The procedures described in this section are listed below. The procedures are located in the public API package PA_RESOURCE_PUB.

- Resource List and Resource List Member Procedures
  - ADD_RESOURCE_LIST_MEMBER, page 3-112
  - CREATE_RESOURCE_LIST, page 3-113
  - DELETE_RESOURCE_LIST, page 3-115
  - DELETE_RESOURCE_LIST_MEMBER, page 3-116
  - SORT_RESOURCE_LIST_MEMBERS, page 3-116
  - UPDATE_RESOURCE_LIST, page 3-114
  - UPDATE_RESOURCE_LIST_MEMBER, page 3-117

- Load-Execute-Fetch Procedures
  - CLEAR_CREATE_RESOURCE_LIST, page 3-118
  - CLEAR_UPDATE_MEMBERS, page 3-118
  - EXEC_CREATE_RESOURCE_LIST, page 3-118
  - EXEC_UPDATE_RESOURCE_LIST, page 3-118
  - FETCH_RESOURCE_LIST, page 3-119
  - FETCH_PLAN_FORMAT
  - FETCH_RESOURCE_LIST_MEMBER, page 3-118
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.

   Note: For more information about resource types in Oracle Projects, see Resource Types, Oracle Projects Fundamentals.

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- PRESOURCE_TYPE_CODE
- PRESOURCE_ATTR_VALUE
- PRESOURCE_ALIAS

**CREATERESOURCE_LIST**

CREATERESOURCE_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 2-22.

**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 PRESOURCE_ATTR_VALUE must correspond with the value for PRESOURCE_TYPE. For example, the person identification code for PRESOURCE_ATTR_VALUE must be valid if PRESOURCE_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.

You can view descriptions of all of the parameters for this procedure in the Oracle
Integration Repository. The Oracle Integration Repository is described in the preface of
this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• RESOURCE_LIST_NAME
• GROUP_RESOURCE_TYPE
• RESOURCE_TYPE_CODE
• RESOURCE_ATTR_VALUE
• RESOURCE_ALIAS

**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 2-22.

**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_BYTYPE 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- RESOURCE_TYPE_CODE
- RESOURCE_ATTR_VALUE
- RESOURCE_ALIAS

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameter for this procedure is listed below:

• P_API_VERSION_NUMBER

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 PRESOURCE_LIST_NAME or the PRESOURCE_LIST_ID. Calling modules can also pass the P_ALIAS or the P_ALIAS_MEMBER_ID.

• If the calling modules pass both PRESOURCE_LIST_NAME and the PRESOURCE_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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER

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 PRESOURCE_LIST_NAME or the PRESOURCE_LIST_ID.
• If the calling modules pass both the PRESOURCE_LIST_NAME and the
  PRESOURCE_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.

You can view descriptions of all of the parameters for this procedure in the Oracle
Integration Repository. The Oracle Integration Repository is described in the preface of
this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_SORT_BY

**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 PRESOURCE_LIST_NAME or
  PRESOURCE_LIST_ID.

• If the calling modules pass both the PRESOURCE_LIST_NAME and the
  PRESOURCE_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.

You can view descriptions of all of the parameters for this procedure in the Oracle
Integration Repository. The Oracle Integration Repository is described in the preface of
this manual.

The required parameter for this procedure is listed below:
• P_API_VERSION_NUMBER
CLEAR_CREATERESOURCELIST

CLEAR_CREATERESOURCELIST is a Load-Execute-Fetch procedure used to clear the global data structures set up during the Initialize step. There are no parameters for this API procedure.

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_CREATERESOURCELIST

EXEC_CREATERESOURCELIST is a Load-Execute-Fetch procedure used to execute the composite API CREATERESOURCELIST.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_RETURN_STATUS

EXEC_UPDATERESOURCELIST

EXEC_UPDATERESOURCELIST is a Load-Execute-Fetch procedure used to execute the composite API UPDATERESOURCELIST.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameter for this procedure is listed below:

- P_API_VERSION_NUMBER

FETCH_MEMBERS

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameter for this procedure is listed below:

- P_API_VERSION_NUMBER

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameter for this procedure is listed below:

- P_API_VERSION_NUMBER

**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 PRESOURCE_LIST_NAME and PRESOURCE_LIST_ID, the API uses the latter.

- If the resource list is grouped, you must pass a valid resource group alias.

- The value for PRESOURCE_ATTR_VALUE must correspond to the value for PRESOURCE_TYPE. For example, person identification code for PRESOURCE_ATTR_VALUE must be valid if PRESOURCE_TYPE equals EMPLOYEE.
Note: For more information about resource types in Oracle Projects, see Resource Types, Oracle Projects Fundamentals.

• 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameter for this procedure is listed below:

• P_API_VERSION_NUMBER

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of
The required parameter for this procedure is listed below:

- P_API_VERSION_NUMBER

Planning Resource List APIs

This section discusses the APIs used in conjunction with planning resource lists.

Planning Resource List API Views

The following table lists the views that provide parameter data for the planning resource list APIs. The information returned in these views can be included in planning resources in Oracle Projects.

For detailed view descriptions, refer to Oracle eTRM, which is available on Oracle MetaLink.

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_EMPLOYEES_RES_V</td>
<td>Information about employees in the human resources system.</td>
</tr>
<tr>
<td>PA_EVENT_TYPES_RES_V</td>
<td>Displays information about event types defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA_EXPEND_CATEGORIES_RES_V</td>
<td>Displays information about expenditure categories defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA_EXPENDITURE_TYPES_RES_V</td>
<td>Displays information about expenditure types defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA_JOBS_RES_V</td>
<td>Displays information about jobs defined in your human resources system.</td>
</tr>
<tr>
<td>PA_ORGANIZATIONS_RES_V</td>
<td>Displays information about organization defined in your human resources system.</td>
</tr>
<tr>
<td>PA_BOM_LABOR_RES_V</td>
<td>Displays information about BOM labor resources defined in your manufacturing system.</td>
</tr>
<tr>
<td>PA_BOM_EQUIPMENT_RES_V</td>
<td>Displays information about BOM equipment resources defined in your manufacturing system.</td>
</tr>
<tr>
<td>PA_ITEM_CATEGORY_RES_V</td>
<td>Displays information about item categories defined in your manufacturing system.</td>
</tr>
</tbody>
</table>
### Planning Resource List API Procedures

The procedures described in this section are listed below. The procedures are located in the public API package PA_PLAN_RES_LIST_PUB.

1. Planning Resource List and Planning Resource List Member Procedures
   - CREATE_RESOURCE_LIST, page 3-131
   - UPDATE_RESOURCE_LIST, page 3-132
   - DELETERESOURCE_LIST, page 3-133
   - DELETE_PLANNINGRESOURCE, page 3-134
   - DELETE_PLAN_RL_FORMAT, page 3-135

2. Load-Execute-Fetch Procedures
   - EXEC_CREATE_RESOURCE_LIST, page 3-135
   - EXEC_UPDATE_RESOURCE_LIST, page 3-136
   - FETCHRESOURCE_LIST, page 3-136

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_ITEMS_RES_V</td>
<td>Displays information about items defined in your manufacturing system.</td>
</tr>
<tr>
<td>PA_NON_LABOR_RESOURCES_RES_V</td>
<td>Displays information about non-labor resources defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA RESOURCE CLASS_RES_V</td>
<td>Displays information about resource classes defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA_PROJECT_ROLES_RES_V</td>
<td>Displays information about project roles defined in Oracle Projects.</td>
</tr>
<tr>
<td>PA_VENDORS_RES_V</td>
<td>Displays information about vendors defined in your Purchasing system.</td>
</tr>
<tr>
<td>PA_PERSON_TYPE_RES_V</td>
<td>Displays information about person types in your human resources system.</td>
</tr>
<tr>
<td>PA_REVENUE_CATEGORIES_RES_V</td>
<td>Displays information about revenue categories defined in Oracle Projects.</td>
</tr>
</tbody>
</table>
• FETCH_PLAN_FORMAT, page 3-136
• FETCH_RESOURCE_LIST_MEMBER, page 3-137
• INIT_CREATERESOURCE_LIST, page 3-137
• INIT_UPDATERESOURCE_LIST, page 3-137
• LOADRESOURCE_LIST, page 3-137
• LOADRESOURCE_FORMAT, page 3-139
• LOAD_PLANNINGRESOURCE, page 3-139

Planning Resource List API Record and Table Datatypes

The record and table datatypes used by the planning resource list APIs are defined on the following pages.

Plan_Res_List_IN_Rec Datatype Parameters

This is the planning resource list record structure. You need to pass the planning resource list record whenever you are creating a planning resource list, or when you are updating an existing planning resource list. You have to pass the default attributes only if you need to modify them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESOURCE_LIST_ID</td>
<td>NUMBER</td>
<td>The resource list identifier.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A value is passed only when you update the resource list.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The value comes from PA_RESOURCE_LISTS_V.</td>
</tr>
<tr>
<td>PRESOURCE_LIST_NAME</td>
<td>VARCHAR2 (80)</td>
<td>The resource list name</td>
</tr>
<tr>
<td>P_DESCRIPTION</td>
<td>VARCHAR2 (255)</td>
<td>The resource list description</td>
</tr>
<tr>
<td>PSTART_DATE</td>
<td>DATE</td>
<td>The resource list start date, passed during resource list creation</td>
</tr>
<tr>
<td>PEND_DATE</td>
<td>DATE</td>
<td>The resource list end date</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_JOB_GROUP_ID</td>
<td>NUMBER</td>
<td>The job group ID of the job associated with the resource list. This value comes from the view PA_JOBS_VIEW.</td>
</tr>
<tr>
<td>P_JOB_GROUP_NAME</td>
<td>VARCHAR2</td>
<td>The job group name of the job associated with the resource list. You can pass either the name or the P_JOB_GROUP_ID value.</td>
</tr>
<tr>
<td>P_USE_FOR_WP_FLAG</td>
<td>VARCHAR2</td>
<td>Flag to indicate whether the resource list can be associated with a workplan. Y indicates that the resource list will be used for workplan. N indicates that the resource list will not be used in workplan.</td>
</tr>
<tr>
<td>P_CONTROL_FLAG</td>
<td>VARCHAR2</td>
<td>Flag to indicate whether the resource list is centrally controlled or project specific. Y indicates that the resource list is centrally controlled. N indicates that the resource list is project specific.</td>
</tr>
<tr>
<td>P_RECORD_VERSION _NUMBER</td>
<td>NUMBER</td>
<td>The record version number of the resource list</td>
</tr>
</tbody>
</table>

**Plan_Res_List_OUT_Rec Data Type Parameters**

This is the planning resource list record structure, which stores the resource list identifier of the newly created planning resource list as an out parameter. The system passes this resource list identifier value when you create resource formats and resource list members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_RESOURCE_LIST_ID</td>
<td>NUMBER</td>
<td>Resource list identifier of any new planning resource list</td>
</tr>
</tbody>
</table>

**Plan_RL_Format_In_Tbl Data Type Parameters**

The record type Plan_RL_Format_In_Tbl is a table of Plan_RL_Format_In_Rec.

**Plan_RL_Format_In_Rec Data Type Parameters**

This is the planning resource list format structure. You must pass the planning resource list format record when you create a resource format. You can add or delete resource formats while updating them. The following table shows the attributes of Plan_RL_Format_In_Rec.
### Plan_RL_Format_Out_Tbl Data Type Parameters

The record type `Plan_RL_Format_Out_Tbl` is a table of `Plan_RL_Format_Out_Rec`. The following table describes attributes of `Plan_RL_Format_Out_Rec`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_PLAN_RL_FORMAT_ID</td>
<td>NUMBER</td>
<td>Identifier of the new planning resource format</td>
</tr>
<tr>
<td>X_RECORD_VERSION_NUMBER</td>
<td>NUMBER</td>
<td>Record version number of the resource format</td>
</tr>
</tbody>
</table>

### Planning_Resource_In_Tbl Data Type Parameters

The record type `Planning_Resource_In_Tbl` is a table of `Planning_Resource_In_Rec`. The following table describes attributes of `Planning_Resource_In_Rec`.

### Planning_Resource_In_Rec Data Type Parameters

This is the planning resource list member record structure. You need to pass the resource list member record whenever you create or update a planning resource list. You must pass the default attributes only if you need to modify or update them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRESOURCE_LIST</td>
<td>NUMBER</td>
<td>Optional resource list member identifier.</td>
</tr>
<tr>
<td>MEMBER_ID</td>
<td>NUMBER</td>
<td>A value is passed when updating a resource list member.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_RESOURCE_ALIAS</td>
<td>VARCHAR2</td>
<td>Alias name of the resource.</td>
</tr>
<tr>
<td>(80)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_PERSON_ID</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of NAMED_PERSON type. It contains the selected resource identifier.</td>
</tr>
<tr>
<td>P_PERSON_NAME</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of NAMED_PERSON type. It contains the person name.</td>
</tr>
<tr>
<td>(240)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_JOB_ID</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of JOB type. It contains job identifier.</td>
</tr>
<tr>
<td>P_JOB_NAME</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of JOB type. It holds the name of the selected resource</td>
</tr>
<tr>
<td>(240)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_ORGANIZATION_ID</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of ORGANIZATION type. It holds the selected resource identifier. This sets the value for ORGANIZATION_ID of PA_RESOURCE_LIST_MEMBERS table.</td>
</tr>
<tr>
<td>P_ORGANIZATION_NAME</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of ORGANIZATION type. It holds the name of the selected resource.</td>
</tr>
<tr>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_VENDOR_ID</td>
<td>NUMBER</td>
<td>The vendor identifier. This sets the value of VENDOR_ID of PA_RESOURCE_LIST_MEMBERS table.</td>
</tr>
<tr>
<td>P_VENDOR_NAME</td>
<td>VARCHAR2</td>
<td>This holds the vendor name.</td>
</tr>
<tr>
<td>(240)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_FIN_CATEGORY_NAME</td>
<td>VARCHAR2</td>
<td>This holds the financial category name.</td>
</tr>
<tr>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_NON_LABOR_NAME</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of NON_LABORRESOURCE type. This holds the non labor resource name.</td>
</tr>
<tr>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_PROJECT_ROLE_ID</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of ROLE type. It holds the selected resource identifier.</td>
</tr>
<tr>
<td>P_PROJECT_ROLE_NAME</td>
<td>VARCHAR2 (80)</td>
<td>This parameter contains a value if the resource is of ROLE type. It holds the name of the selected resource.</td>
</tr>
<tr>
<td>P_RESOURCE_CLASS_ID</td>
<td>NUMBER</td>
<td>This parameter contains the identifier of the resource class to which the resource belongs. It can take the following values: 1, 2, 3, or 4.</td>
</tr>
<tr>
<td>PRESOURCE_CLASS_CODE</td>
<td>VARCHAR2 (30)</td>
<td>This parameter contains the code of the resource class to which the resource belongs. It can take the following values: EQUIPMENT, FINANCIAL ELEMENTS, MATERIAL ITEMS PEOPLE</td>
</tr>
<tr>
<td>P_RES_FORMAT_ID</td>
<td>NUMBER</td>
<td>It should be passed during creation of a planning resource list member. This holds the planning resource format ID to which the resource belongs.</td>
</tr>
<tr>
<td>P_SPREAD_CURVE_ID</td>
<td>NUMBER</td>
<td>Optional, defines the way cost or revenue amounts are distributed across periods for financial planning. This holds the identifier of the spread curves available. It sets the value for SPREAD_CURVE_ID of PARESOURCE_LIST_MEMBERS table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If P_SPREAD_CURVE_ID is NULL, then SPREAD_CURVE_ID is set to its default value decided by the resource class passed in.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If P_SPREAD_CURVE_ID is NOT NULL, then SPREAD_CURVE_ID is set to whatever is passed.</td>
</tr>
<tr>
<td>P_ETC_METHOD_CODE</td>
<td>VARCHAR2 (30)</td>
<td>Optional, users can setup Estimate to Complete (ETC) Methods by resource. This planning attribute holds the corresponding ETC code. P_ETC_METHOD_CODE sets the value for ETC_METHOD_CODE of PARESOURCE_LIST_MEMBERS table. If P_ETC_METHOD_CODE is NULL then ETC_METHOD_CODE is set to its default value decided by the resource class passed in. If P_ETC_METHOD_CODE is NOT NULL then ETC_METHOD_CODE is set to whatever is passed.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_MFC_COST_TYPE_ID</td>
<td>NUMBER</td>
<td>Optional, holds the manufacturing cost type identifier. It sets the value for MFC_COST_TYPE_ID of PARESOURCE_LIST_MEMBERS table with whatever is passed in P_MFC_COST_TYPE_ID. If P_MFC_COST_TYPE_ID is NULL and the resource type is either BOM_EQUIPMENT or BOM_LABOR or INVENTORY_ITEM then MFC_COST_TYPE_ID is set to its default value decided by the resource class else MFC_COST_TYPE_ID is set to NULL.</td>
</tr>
<tr>
<td>P_COPY_FROM_RL_FLAG</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>P_RESOURCE_CLASS_FLAG</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>P_FC_RES_TYPE_CODE</td>
<td>VARCHAR2</td>
<td>Decides the value for WP_ELIGIBLE_FLAG of PARESOURCE_LIST_MEMBERS table. Based on this code WP_ELIGIBLE_FLAG is set to either ‘Y’ or ‘N’. If P_FC_RES_TYPE_CODE is either ‘REVENUE_CATEGORY’ or ‘EVENT_TYPE’ then WP_ELIGIBLE_FLAG will hold ‘N’ else ‘Y’.</td>
</tr>
<tr>
<td>P_INVENTORY_ITEM_ID</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of ITEM type. It contains the resource identifier.</td>
</tr>
<tr>
<td>P_INVENTORY_ITEM_NAME</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of ITEM type. It holds the selected resource name.</td>
</tr>
<tr>
<td>P_ITEM_CATEGORY_ID</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of ITEM CATEGORY type. It holds the selected resource identifier.</td>
</tr>
<tr>
<td>P_ITEM_CATEGORY_NAME</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of ITEM CATEGORY type. It holds the selected resource name.</td>
</tr>
<tr>
<td>P_MIGRATION_CODE</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>-------------</td>
</tr>
<tr>
<td>P_ATTRIBUTE</td>
<td>VARCHAR2</td>
<td>The name of the attribute category</td>
</tr>
<tr>
<td>_CATEGORY</td>
<td>(150)</td>
<td></td>
</tr>
<tr>
<td>P_ATTRIBUTE1 TO P_ATTRIBUTE30</td>
<td>VARCHAR2</td>
<td>The descriptive flexfields enable users to define additional information for each planning resource</td>
</tr>
<tr>
<td>_CODE</td>
<td>(150)</td>
<td></td>
</tr>
<tr>
<td>P_PERSON_TYPE</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of PERSON_TYPE type. It is the person type code or name.</td>
</tr>
<tr>
<td>_CODE</td>
<td>(30)</td>
<td></td>
</tr>
<tr>
<td>P_BOMRESOURCE</td>
<td>NUMBER</td>
<td>This parameter contains a value if the resource is of BOM_LABOR or BOM_EQUIPMENT type. It holds the selected resource identifier.</td>
</tr>
<tr>
<td>_ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_BOMRESOURCE</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of BOM_LABOR or BOM_EQUIPMENT type. It holds the selected resource name.</td>
</tr>
<tr>
<td>_NAME</td>
<td>(30)</td>
<td></td>
</tr>
<tr>
<td>P_TEAM_ROLE</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource is of NAMED_ROLE type. It holds the team role name or code.</td>
</tr>
<tr>
<td>(80)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_INCURBYRES</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource belongs to a format having Incurred By Resource as planning resource element. It holds the selected resource code.</td>
</tr>
<tr>
<td>_CODE</td>
<td>(30)</td>
<td></td>
</tr>
<tr>
<td>P_INCURBYRES</td>
<td>VARCHAR2</td>
<td>This parameter contains a value if the resource belongs to a format having Incurred By Resource as planning resource element.</td>
</tr>
<tr>
<td>_TYPE</td>
<td>(30)</td>
<td></td>
</tr>
<tr>
<td>P_RECORD</td>
<td>NUMBER</td>
<td>This required attribute is the record version number of the resource list member. It has significance during the update of the resource list member.</td>
</tr>
<tr>
<td>_VERSION_NUMBER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_PROJECT_ID</td>
<td>NUMBER</td>
<td>Required, holds the project ID for project specific resources. It determines the value for OBJECT_TYPE and OBJECT_ID of PA_RESOURCE_LIST_MEMBERS table. If PROJECT_ID is NOT NULL, OBJECT_TYPE and OBJECT_ID takes values PROJECT and PROJECT_ID. If PROJECT_ID is NULL, then it takes RESOURCE_LIST and P_RESOURCE_LIST_ID as OBJECT_TYPE and OBJECT_ID.</td>
</tr>
</tbody>
</table>
### Name Type Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_ENABLED_FLAG</td>
<td>VARCHAR2</td>
<td>This optional attribute indicates whether or not the resource member is enabled. The value need not be passed unless you want to modify its value during update. ENABLED_FLAG will always be Y during creation of a resource list member. Y = The resource list member is enabled. N = The resource list member is disabled.</td>
</tr>
</tbody>
</table>

#### Planning_Resource_Out_Tbl Data Type Parameters

The record type `Planning_Resource_Out_Tbl` is a table of `Planning_Resource_Out_Rec`. The following table describes attributes of `Planning_Resource_Out_Rec`.

#### Planning_Resource_Out_Rec Data Type Parameters

This is the planning resource list member record structure, which stores the resource list member identifier of the new resource list member as an out parameter.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRESOURCE_LIST_MEMBER_ID</td>
<td>NUMBER</td>
<td>Identifier of the new resource list member</td>
</tr>
<tr>
<td>X_RECORD_VERSION_NUMBER</td>
<td>NUMBER</td>
<td>Record version number of the resource list member</td>
</tr>
</tbody>
</table>

#### Planning Resource List API Definitions

This section contains descriptions of the planning resource list APIs, including business rules and parameters.

**CREATE_RESOURCE_LIST**

CREATE_RESOURCE_LIST is a PL/SQL procedure that enables users to create resource lists, corresponding resource formats, and resource list members.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes, page 2-22.

**Business Rules**

- CREATE_RESOURCE_LIST validates the resource list parameters and creates the
If the system creates the resource list successfully, it then creates resource formats and passes them as parameters to CREATE_RESOURCE_LIST.

If the system cannot add a resource format to the resource list, the API fails.

If the system adds the resource formats successfully, it then processes the planning resources. If it fails to create any planning resource, the API fails.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PLAN_RES_LIST_REC
- P_PLAN_RL_FORMAT_TBL
- P_PLANNING_RESOURCE_IN_TBL

**UPDATE_RESOURCE_LIST**

UPDATE_RESOURCE_LIST is a PL/SQL procedure that enables you to update a resource list, corresponding resource formats, and planning resources. Since a resource format cannot be updated, it creates a resource format. It can also enable you to update existing planning resources or create a new planning resource if no planning resource exists.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes, page 2-22.

**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 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 add new resource formats to the planning resource list by passing the formats P_PLAN_RL_FORMAT_TBL.

• You can update existing or add new planning resources by passing the planning resources in P_PLANNINGRESOURCE_IN_REC.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER

• P_PLAN_RES_LIST_REC

• X_PLAN_RES_LIST_REC

• P_PLAN_RL_FORMAT_TBL

• X_PLAN_RL_FORMAT_TBL

• P_PLANNINGRESOURCE_IN_TBL

• X_PLANNINGRESOURCE_OUT_TBL

• X_MSG_COUNT

• X_MSG_DATA

• X_RETURN_STATUS

DELETE_RESOURCE_LIST

DELETE_RESOURCE_LIST is a PL/SQL procedure that enables you to delete a resource list and corresponding planning resources and resource formats belonging to the resource list.

Business Rules

• Calling modules can pass either the P_RESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID.

• When the planning resource list is deleted, all the resource formats and planning
resources belonging to that resource are also deleted.

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

- You cannot delete the planning resource list if it is referenced by a financial plan or workplan.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_RES_LIST_ID
- X_MSG_COUNT
- X_MSG_DATA
- X_RETURN_STATUS

DELETE_PLANNING RESOURCE

DELETE_PLANNING_RESOURCE is a PL/SQL procedure that enables you to delete a planning resource that is not in use or disable a planning resource that is in use.

Business Rule

The calling module passes a table containing valid resource list member IDs. This procedure deletes each list member if it is not in use, or disables it if it is in use. If the list member ID does not exist, the procedure does not raise an error message, and returns a Success result.

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- PRESOURCE_LIST_MEMBER_ID
- X_RETURN_STATUS
- X_MSG_COUNT
**DELETE_PLAN_RL_FORMAT**

This procedure deletes one or more resource formats from a resource list.

**Business Rule**

The calling module has to pass in a table of valid resource format IDs and the ID of the resource list they belong to. The API will delete each format from the resource list if there are no planning resources of that format on the list. If any planning resources exist, the resource format cannot be deleted from the list.

**Parameters**

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_RES_LIST_ID
- P_PLAN_RL_FORMAT_TB
- X_RETURN_STATUS
- X_MSG_COUNT
- X_ERROR_MSG_DATA

**EXEC_CREATE_RESOURCE_LIST**

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- X_RETURN_STATUS
- X_MSG_COUNT
- X_MSG_DATA
EXEC_UPDATE_RESOURCE_LIST

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• X_RETURN_STATUS
• X_MSG_COUNT
• X_MSG_DATA

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. It returns the status and the new RESOURCE_LIST_ID, if any.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• X_RETURN_STATUS
• X_RESOURCE_LIST_ID
• X_LIST_RETURN_STATUS

FETCH_PLAN_FORMAT

FETCH_PLAN_FORMAT is a Load-Execute-Fetch procedure that returns the planning resource format status and the new PLAN_RL_FORMAT_ID.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• X_RETURN_STATUS
• X_PLAN_RL_FORMAT_ID
• X_FORMAT_RETURN_STATUS

**FETCHRESOURCELISTMEMBER**

FETCHRESOURCELISTMEMBER is a Load-Execute-Fetch procedure that returns the resource list member status and the new RESOURCE_LIST_MEMBER_ID, if any exists.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• X_RETURN_STATUS
• X_RESOURCE_LIST_MEMBER_ID
• X_MEMBER_RETURN_STATUS

**INITCREATERESOURCELIST**

INITCREATERESOURCELIST is a Load-Execute-Fetch procedure used to set up the global temporary tables for resource formats and resource list members. It also initializes the record structure for the resource list. There are no parameters for this procedure.

**INITUPDATERESOURCELIST**

INITUPDATERESOURCELIST is a Load-Execute-Fetch procedure used to initialize the global temporary tables for the resource formats and resource list members. It also initializes the record structure for the resource list. There are no parameters for this procedure.

**LOADRESOURCELIST**

LOADRESOURCELIST is a Load-Execute-Fetch procedure that enables you to load the global record for a resource list with values.
Business Rules

- The following parameters must be passed when you create a new planning resource list:
  - P_RESOURCE_LIST_NAME
  - P_START_DATE

- The following parameters are optional when you create a new planning resource list:
  - P_DESCRIPTION
  - P_END_DATE
  - P_JOB_GROUP_ID
  - P_JOB_GROUP_NAME
  - P_USE_FOR_WP_FLAG
  - P_CONTROL_FLAG

- The following parameters can be updated when you update a planning resource list:
  - P_RESOURCE_LIST_NAME
  - P_START_DATE
  - P_DESCRIPTION
  - P_END_DATE
  - P_JOB_GROUP_ID
  - P_JOB_GROUP_NAME
  - P_USE_FOR_WP_FLAG
  - P_CONTROL_FLAG

- To identify which resource needs to be updated, you should pass the PRESOURCE_LIST_ID identifier, which you can get from the PA_RESOURCE_LISTS_V view.

You can view descriptions of all of the parameters for this procedure in the Oracle
The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- PRESOURCE_LIST_NAME
- PDESCRIPTION
- PJOB_GROUP_ID
- X_RETURN_STATUS

LOAD_RESOURCE_FORMAT

Enables you to load values into the global table for resource formats. This procedure populates the table and increments the count.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- X_RETURN_STATUS

LOAD_PLANNING_RESOURCE

Enables the user to load the global table for planning resources. This procedure populates the table and increments the count.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_RES_FORMAT_ID
- P_RECORD_VERSION_NUMBER
- P_PROJECT_ID
Resource Breakdown Structure APIs

This section discusses the APIs used in conjunction with resource breakdown structures. They enable you to create resource breakdown structures, assign them to projects, and update them.

Resource Breakdown Structure API Views

The following table lists the views that provide parameter data for the resource breakdown structure APIs. The information returned in these views can be included in resource breakdown structures in Oracle Projects.

For detailed description of the views, refer to Oracle eTRM, which is available on Oracle MetaLink.

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_EMPLOYEES_RES_V</td>
<td>Displays information about employees in the Human Resources system</td>
</tr>
<tr>
<td>PA_EVENT_TYPES_RES_V</td>
<td>Displays information about event types defined in Oracle Projects</td>
</tr>
<tr>
<td>PA_EXPEND_CATEGORIES_RES_V</td>
<td>Displays information about expenditure categories defined in Oracle projects</td>
</tr>
<tr>
<td>PA_EXPENDITURE_TYPES_RES_V</td>
<td>Displays information about expenditure types defined in Oracle projects</td>
</tr>
<tr>
<td>PA_JOBS_RES_V</td>
<td>Displays information about jobs defined in your Human resources system.</td>
</tr>
<tr>
<td>PA_ORGANIZATIONS_RES_V</td>
<td>Displays information about organization defined in your Human resources system.</td>
</tr>
<tr>
<td>PA_BOM_LABOR_RES_V</td>
<td>Displays information about BOM labor resources defined in your manufacturing system</td>
</tr>
<tr>
<td>PA_BOM_EQUIPMENT_RES_V</td>
<td>Displays information about BOM equipment resource defined in your manufacturing system</td>
</tr>
<tr>
<td>PA_ITEMCATEGORY_RES_V</td>
<td>Displays information about item categories defined in your manufacturing system</td>
</tr>
<tr>
<td>PA_ITEMS_RES_V</td>
<td>Displays information about items defined in your manufacturing system</td>
</tr>
</tbody>
</table>
### Resource Breakdown Structure API Procedures

The procedures described in this section are listed below. The procedures are located in the public API package PA_RBS_PUB.

- Resource Breakdown Structure Procedures
  - CREATE_RBS, page 3-147
  - COPY_RBS_WORKING_VERSION, page 3-148
  - UPDATE_RBS, page 3-149
  - INIT_RBS_PROCESSING, page 3-150
  - LOAD_RBS_HEADER, page 3-150
Resource Breakdown Structure API Record and Table Datatypes

The record and table datatypes used by resource breakdown structure APIs are defined on the following pages.

**Rbs_Header_Rec_Typ Datatype Parameters**

This is the header record structure of the resource breakdown structure. When create or update a resource breakdown structure header, you must pass the resource breakdown structure. You only pass the default attributes if you need to modify them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| RBS_HEADER_ID    | NUMBER         | The header identifier of the resource breakdown structure.  
<pre><code>              |                | The value is passed when the header is updated.          |
</code></pre>
<p>| NAME             | VARCHAR2 (240) | Resource breakdown structure name.               |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EFFECTIVE_FROM_DATE</td>
<td>DATE</td>
<td>The date from which the resource breakdown structure can be used. Resource breakdown structures with effective from dates that are before the current date are eligible to be assigned to projects. EFFECTIVE_FROM_DATE cannot be NULL.</td>
</tr>
<tr>
<td>EFFECTIVE_TO_DATE</td>
<td>DATE</td>
<td>The date up to which the resource breakdown structure can be used. Resource breakdown structures with effective to dates that are on or after the current date are eligible to be assigned to projects.</td>
</tr>
<tr>
<td>EFFECTIVE_TO_DATE</td>
<td>NUMBER</td>
<td>The record version number of the resource breakdown structure header record (from the PA_RBS_HEADERS_AMG_V view).</td>
</tr>
</tbody>
</table>

**Rbs_Version_Rec_Typ Datatype Parameters**

This is the version record structure for the resource breakdown structure. When you create a resource breakdown structure, you pass the version record only if you want any specific version attributes. By default, the system creates a working version whenever you are creating a resource breakdown structure header. When you update a resource breakdown structure, you pass the version record only if you want to update version attributes such as version name or job group. You can update only the current working version. The version attributes can be read from the PA_RBS_VERSIONS_AMG_V view.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBS_VERSION_ID</td>
<td>NUMBER</td>
<td>The version identifier of the resource breakdown structure (RBS).</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR2 (240)</td>
<td>Resource breakdown structure version name</td>
</tr>
<tr>
<td>VERSION_START_DATE</td>
<td>DATE</td>
<td>The resource breakdown structure effective start date. The system uses this date to decide whether all the project transactions should be mapped to the version. If the date is before the system date, the system maps the transactions to this version.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JOB_GROUP_ID</td>
<td>NUMBER</td>
<td>The job group identifier of the job group for the resource breakdown structure version. Jobs from this job group are eligible to be elements of the resource breakdown structure hierarchy. You can select the job group from the PA_JOB_GROUPS_VIEW view.</td>
</tr>
<tr>
<td>RECORD_VERSION_NUMBER</td>
<td>NUMBER</td>
<td>The record version number of the RBS version record from the PA_RBS_VERSIONS_AMG_V view.</td>
</tr>
</tbody>
</table>

**Rbs_Elements_Tbl_Typ Datatype Parameters**

The record type `Rbs_Elements_Tbl_Typ` is a table of `Rbs_Elements_Rec_Typ`. The following table describes attributes of `Rbs_Elements_Rec_Typ`.

**Rbs_Elements_Rec_Typ Datatype Parameters**

This is the element record structure for the resource breakdown structure. Whenever you create or update a resource breakdown structure element, you pass the element record structure. You pass the default attributes only if you need to modify or update them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBS_VERSION_ID</td>
<td>NUMBER</td>
<td>The resource breakdown structure’s version identifier.</td>
</tr>
<tr>
<td>RBS_ELEMENT_ID</td>
<td>NUMBER</td>
<td>The resource breakdown structure’s element’s identifier. This is a required field in the update mode.</td>
</tr>
<tr>
<td>PARENT_ELEMENT_ID</td>
<td>NUMBER</td>
<td>The current element parent’s element identifier.</td>
</tr>
<tr>
<td>RESOURCE_TYPE_ID</td>
<td>NUMBER</td>
<td>The resource type identifier of the element. You can get the value from PA_RES_TYPES_AMG_V view.</td>
</tr>
<tr>
<td>RESOURCE_SOURCE_ID</td>
<td>NUMBER</td>
<td>The resource identifier that makes up the resource breakdown structure element.</td>
</tr>
<tr>
<td>RESOURCE_SOURCE_CODE</td>
<td>VARCHAR2 (240)</td>
<td>The source code of the resource breakdown structure element. It has a value if the resource type of the element is associated with the resource type code of REVENUE_CATEGORY or USER_DEFINED.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ORDER_NUMBER</td>
<td>NUMBER</td>
<td>The order in which the elements should be displayed on a given level of the resource breakdown structure in project reporting.</td>
</tr>
<tr>
<td>PROCESS_TYPE</td>
<td>VARCHAR 2</td>
<td>Type of processing required for the resource breakdown structure element. The valid values are:</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td>A Add element</td>
</tr>
<tr>
<td></td>
<td></td>
<td>U Update element</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D Delete element and its children if the element exists</td>
</tr>
<tr>
<td>RBS_LEVEL</td>
<td>NUMBER</td>
<td>Level at which the element is placed in the resource breakdown structure. This value is passed when the element is created. The level can have a value between 1 and 10. 1 is reserved for the root element, which will be ignored during processing.</td>
</tr>
<tr>
<td>RECORD_VERSION_NUMBER</td>
<td>NUMBER</td>
<td>Record version number of the resource breakdown structure's element. It is passed when the element is updated or deleted. You can get the value from the PA_RBS_ELEMENTS_AMG_V view.</td>
</tr>
<tr>
<td>RBS_REF_ELEMENT_ID</td>
<td>NUMBER</td>
<td>Identifier used to distinguish each element processed at each RBS level. It is passed when the element is created. Each element must have a RBS_REF_ELEMENT_ID.</td>
</tr>
<tr>
<td>PARENT_REF_ELEMENT_ID</td>
<td>NUMBER</td>
<td>Indicates the element parent. If the RBS_LEVEL is 1 or 2 then it does not need to be populated. RBS_LEVEL 1 elements do not need parents. The RBS_LEVEL 2 elements parents are updated by the system from the root element's ID.</td>
</tr>
</tbody>
</table>

**RESOURCE_SOURCE_ID**

The RESOURCE_SOURCE_ID parameter is populated with the resource identifier that makes up the resource breakdown structure element. If the element is a rule, then the system passes -1 as the RESOURCE_SOURCE_ID. If the element is an instance, the system passes the resource identification code.

Following are the resource types with identification codes:

- **BOM_LABOR**: Get the BOM Labor ID from the PA_BOM_LABOR_RES_V view
- **BOM_EQUIPMENT**: Get the BOM Equipment ID from the PA_BOM_EQUIPMENT_RES_V view
- **NAMED_PERSON**: Get the Person ID from the PA_EMPLOYEES_RES_V view
- **EVENT_TYPE**: Get the Event ID from the PA_EVENT_TYPES_RES_V view
- **EXPENDITURE_CATEGORY**: Get the Expenditure Category ID from the PA_EXPEND_CATEGORIES_RES_V view
- **EXPENDITURE_TYPE**: Get the Expenditure Type ID from the PA_EXPENDITURE_TYPES_RES_V view
- **ITEMCATEGORY**: Get the Item Category ID from the PA.ITEMCATEGORY_RES_V view
- **INVENTORY_ITEM**: Get the Item ID from the PA_ITEMS_RES_V view
- **JOB**: Get the job ID from the PA_JOBS_RES_V view
- **ORGANIZATION**: Get the Organization ID from the PA.ORGANIZATIONS_RES_V view
- **NON_LABOR_RESOURCE**: Get the Non-Labor Resource ID from the PA_NON_LABOR_RESOURCES_RES_V view
- **RESOURCE_CLASS**: Get the Resource Class ID from PA.getResource_CLASS_RES_V
- **ROLE**: Get the Project Role ID from the PA_PROJECT_ROLES_RES_V view
- **SUPPLIER**: Get the Vendor ID from the PA_VENDORS_RES_V view

**Rbs_Elements_Tbl_Out_Typ Datatype Parameters**

The record type **Rbs_Elements_Tbl_Out_Typ** is a table of Rbs_Elements_Rec_Out_Typ.

**Rbs_Elements_Rec_Out_Typ Datatype Parameters**

This is the element record structure for the resource breakdown structure that stores the output values of new resource breakdown structure elements. The following table describes attributes of **Rbs_Elements_Rec_Out_Typ**.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBS_ELEMENT_ID</td>
<td>NUMBER</td>
<td>The resource breakdown structure's element identifier for the new element</td>
</tr>
</tbody>
</table>
Resource Breakdown Structure API Procedure Definitions

This section contains descriptions of the resource breakdown structure APIs, including business rules and parameters.

CREATE_RBS

The CREATE_RBS procedure enables you to create a resource breakdown structure (RBS), which is composed of the RBS header, the RBS version, and it's elements of the hierarchy.

Business Rules

- CREATE_RBS creates a resource breakdown structure. At minimum, CREATE_RBS creates a header and a working version for a resource breakdown structure.

- If the version record is not populated, CREATE_RBS uses the data stored in the P_HEADER_REC parameter to create the version record.

- If the version information is passed, the version start date must be greater or equal to the effective from date of the header record.

- CREATE_RBS can also create the entire resource breakdown structure hierarchy if the elements of the hierarchy are passed.

- The root element of the hierarchy of a resource breakdown structure version is the version itself. The version is not an updatable element. The system ignores any alternative elements that you may provide.

- The following information must be passed for each element in the resource breakdown structure hierarchy:
  - RBS_LEVEL
  - PROCESS_TYPE
  - PARENT_RES_ELEMENT_ID
  - RESOURCE_TYPE_ID
  - RESOURCE_SOURCE_ID or RESOURCE_SOURCE_CODE

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_HEADER_REC
• X_RBS_HEADER_ID
• X_RBS_VERSION_ID
• X_ELEMENTS_TBL
• X_RETURN_STATUS
• X_MSG_COUNT
• X_ERROR_MSG_DATA

COPY_RBS_WORKING_VERSION

This procedure enables you to create a working version of a resource breakdown structure from an existing frozen version.

Business Rules

1. The calling module must pass one of the following:
   • a resource breakdown structure header (Name or ID) and the version number of a frozen resource breakdown structure version
   • the version ID of a frozen resource breakdown structure version

   This information specifies version that will be copied as the current working version. The existing working version will be overwritten by a copy of the specified frozen version.

2. The record version number of the current working version is also required for locking purposes.

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_REC_VERSION_NUMBER
• X_RETURN_STATUS
• X_MSG_COUNT
• X_ERROR_MSG_DATA

**UPDATE_RBS**

The UPDATE_RBS procedure enables you to update the resource breakdown structure header and version and delete, update, or add records to element records.

**Business Rules**

• UPDATE_RBS can be used to update the resource breakdown structure version, header, or elements, individually or in combination with each other.

• If P_HEADER_REC, P_VERSION_REC, and P_ELEMENTS_TBL are null then nothing is updated.

• If UPDATE_RBS is called for a nonexistent resource breakdown structure, it returns a “No Data Found” error message and stops.

• If the P_ELEMENTS_TBL is populated then the data will be processed in the following order: PROCESS_TYPE 'D,' 'U,' 'A.'

• You cannot delete the root node/element record for UPDATE_RBS because it is system-generated.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER

• P_HEADER_REC

• P_VERSION_REC

• P_ELEMENTS_TBL

• X_ELEMENTS_TBL

• X_RETURN_STATUS

• X_MSG_COUNT
• X_ERROR_MSG_DATA

INIT_RBS_PROCESSING

INIT_RBS_PROCESSING is a Load-Execute-Fetch procedure that initializes the global temporary tables for creating and updating the resource breakdown structure.

LOAD_RBS_HEADER

LOAD_RBS_HEADER is a load-execute-fetch procedure that loads header record information into the global PL/SQL record for the resource breakdown structure.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_NAME
• P_EFFECTIVE_START_DATE
• X_RETURN_STATUS

LOAD_RBS_VERSION

LOAD_RBS_VERSION is a load-execute-fetch procedure that loads version record information into a global PL/SQL record.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_NAME
• P_VERSION_START_DATE
• X_RETURN_STATUS

LOAD_RBS_ELEMENTS

LOAD_RBS_ELEMENTS is a load-execute-fetch procedure that loads element record information into a global PL/SQL table.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- **P_API_VERSION_NUMBER**
- **P_PARENT_ELEMENT_ID**
- **P_RESOURCE_TYPE_ID**
- **P_PROCESS_TYPE**
- **X_RETURN_STATUS**

**FETCH_RBS_HEADER**

FETCH_RBS_HEADER is a load-execute-fetch procedure that retrieves the RBS_HEADER_ID and the header success or error status back to the calling procedure.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- **P_API_VERSION_NUMBER**
- **X_RBS_HEADER_ID**
- **X_RETURN_STATUS**

**FETCH_RBS_VERSION**

FETCH_RBS_VERSION is a load-execute-fetch procedure that retrieves the RBS_VERSION_ID and the version success status.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- **P_API_VERSION_NUMBER**
- **X_RBS_VERSION_ID**
- **X_RBS_VER_RETURN_STATUS**
FETCH_RBS_ELEMENT

FETCH_RBS_ELEMENT is a load-execute-fetch procedure that retrieves the RBS_ELEMENT_ID and the success value for the index value you enter.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_RBS_ELEMENT_INDEX
- X_RBS_ELEMENT_ID
- X_RETURN_STATUS

EXEC_CREATE_RBS

EXEC_CREATE_RBS is a load-execute-fetch procedure that executes the resource breakdown structure creation process.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- X_RETURN_STATUS
- X_MSG_COUNT
- X_ERROR_MSG_DATA

EXEC_UPDATE_RBS

EXEC_UPDATE_RBS is a load-execute-fetch procedure that executes the resource breakdown structure update process.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
FREEZE_RBS_VERSION

FREEZE_RBS_VERSION is a process that freezes the current working resource breakdown structure version and enables the user to create a new working version.

Business Rules

• To freeze a working resource breakdown structure version, you must pass one or more of the following parameters
  • RBS_VERSION_ID
  • RBS_HEADER_ID
  • RBS_HEADER_NAME

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_RBS_VERSION_ID
• P_RBS_VERSION_RECORD_VER_NUM
• X_RETURN_STATUS
• X_MSG_COUNT
• X_ERROR_MSG_DATA

ASSIGN_RBS_TO_PROJECT

ASSIGN_RBS_TO_PROJECT is a process that assigns the resource breakdown structure to a project. You must provide the resource breakdown structure Header ID and the Project ID.
Business Rules

- You must provide the resource breakdown structure and the project to which the resource breakdown structure must be associated.

- To pass project information, you can pass either of the following:
  - the project identifier from the P_PROJECT_ID field
  - the project reference from the P_PROJECT_REFERENCE field

- To pass resource breakdown structure information, you can pass either of the following:
  - the resource breakdown structure header identifier in the P_RBS_HEADER_ID field.
  - the resource breakdown structure header name in the P_RBS_HEADER_NAME field.

- You can assign a resource breakdown structure for the purpose of project reporting and/or program reporting.

- You can also indicate whether the resource breakdown structure should be the primary reporting resource breakdown structure for your project.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PROJECT_ID
- P_RBS_HEADER_ID
- X_RETURN_STATUS
- X_MSG_COUNT
- X_ERROR_MSG_DATA
Dependency APIs

Dependency API Views

The following list shows the views that provide parameter data for the dependency APIs. For detailed description of the views, refer to Oracle eTRM, which is available on OracleMetaLink.

- **PA_STRUCT_TASKS_AMG_V** Displays a list of structure versions
- **PA_TASKS_AMG_V** Displays a list of tasks

Dependency API Procedures

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

- **CREATE_DEPENDENCY**, page 3-155
- **UPDATE_DEPENDENCY**, page 3-156
- **DELETE_DEPENDENCY**, page 3-157

Dependency API Procedure Definitions

This section contains descriptions of the dependency APIs.

**CREATE_DEPENDENCY**

CREATE_DEPENDENCY is a PL/SQL procedure that creates an intra-project dependency.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- **P_MSG_COUNT**
- **P_MSG_DATA**
- **P_RETURN_STATUS**
- **P_PM_PRODUCT_CODE**
UPDATE_DEPENDENCY

UPDATE_DEPENDENCY is a PL/SQL procedure that updates an existing intra-project dependency.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_MSG_COUNT
- P_MSG_DATA
- P_RETURN_STATUS
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
- P_PA_PROJECT_ID
- P_STRUCTURE_VERSION_ID
- P_PA_TASK_ID
- P_PA_PRED_ID
- P_PM_TASK_REFERENCE
- P_PM_PRED_REFERENCE
- P_PA_TASK_ID
DELETE_DEPENDENCY

DELETE_DEPENDENCY is a PL/SQL procedure that deletes an existing intra-project dependency.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_MSG_COUNT
- P_MSG_DATA
- P_RETURN_STATUS
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
- P_PA_PROJECT_ID
- P_STRUCTURE_VERSION_ID
- P_PM_TASK_REFERENCE
- P_PA_TASK_ID
- P_PM_PRED_REFERENCE
- P_PA_PRED_ID

Task Assignment APIs

The task assignment APIs provide an interface for external systems to insert, update, and delete task assignments and periodic data.

Task Assignment Views

The following list shows the views that provide parameter data for the task assignment APIs. For detailed description of the views, refer to Oracle eTRM, which is available on OracleMetaLink.

- PA_TASK_ASSIGNMENTS_A
- MG_V

You can use this view to retrieve valid task assignments from Oracle Projects and display them in your external system.
Task Assignment API Procedures

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

- LOAD_TASK_ASSIGNMENTS, page 3-163
- LOAD_TASK_ASGMT_PERIODS, page 3-163
- EXECUTE_CREATE_TASK_ASGMTS, page 3-163
- EXECUTE_UPDATE_TASK_ASGMTS, page 3-164
- CREATE_TASK_ASSIGNMENTS, page 3-164
- CREATE_TASK_ASSIGNMENT_PERIODS, page 3-164
- UPDATE_TASK_ASSIGNMENTS, page 3-165
- DELETE_TASK_ASSIGNMENTS, page 3-166
- UPDATE_TASK_ASSIGNMENT_PERIODS, page 3-165
- FETCH_TASK_ASSIGNMENTS, page 3-166
- CONVERT_PM_TAREF_TO_ID, page 3-167
- INIT_TASK_ASSIGNMENTS, page 3-167

Task Assignment API Record and Table Datatype

The record and table datatypes used by the task assignment APIs are defined on the following pages.

ASSIGNMENT_IN_REC_TYPE

The following table shows the ASSIGNMENT_IN_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_PROJECT_REFERENCE</td>
<td>VARCHAR2(25)</td>
<td>No</td>
<td>External project reference</td>
</tr>
<tr>
<td>PA_PROJECT_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the project</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_STRUCTURE_VERSION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the structure version</td>
</tr>
<tr>
<td>PM_TASK_REFERENCE</td>
<td>VARCHAR2(25)</td>
<td>No</td>
<td>External task reference</td>
</tr>
<tr>
<td>PA_TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task</td>
</tr>
<tr>
<td>PA_TASK_ELEMENT_VERSION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task version</td>
</tr>
<tr>
<td>PM_TASK_ASGMT_REFERENCE</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>External task assignment reference</td>
</tr>
<tr>
<td>PA_TASK_ASSIGNMENT_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task assignment</td>
</tr>
<tr>
<td>RESOURCE_ALIAS</td>
<td>VARCHAR2(80)</td>
<td>No</td>
<td>Alias of the planning resource</td>
</tr>
<tr>
<td>RESOURCE_LIST_MEMBER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the planning resource</td>
</tr>
<tr>
<td>START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Start date of the task assignment</td>
</tr>
<tr>
<td>END_DATE</td>
<td>DATE</td>
<td>No</td>
<td>End date of the task assignment</td>
</tr>
<tr>
<td>PLANNED_QUANTITY</td>
<td>NUMBER</td>
<td>No</td>
<td>Planned effort or quantity</td>
</tr>
<tr>
<td>PLANNED_TOTAL_RAW_COST</td>
<td>NUMBER</td>
<td>No</td>
<td>Planned raw cost</td>
</tr>
<tr>
<td>PLANNED_TOTAL_BUR_COST</td>
<td>NUMBER</td>
<td>No</td>
<td>Planned burdened cost</td>
</tr>
<tr>
<td>CURRENCY_CODE</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>Currency code</td>
</tr>
<tr>
<td>ATTRIBUTE_CATEGORY</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>Descriptive Flexfield attribute</td>
</tr>
<tr>
<td>ATTRIBUTE1 - ATTRIBUTE30</td>
<td>VARCHAR2(150)</td>
<td>No</td>
<td>Descriptive Flexfield attribute</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(240)</td>
<td>No</td>
<td>Description of the assignment. This parameter can only be used on update.</td>
</tr>
<tr>
<td>USE_TASK_SCHEDULE_FLAG</td>
<td>VARCHAR2(1)</td>
<td>No</td>
<td>Flag indicating whether the assignment dates are the same as the task</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>scheduled dates. This parameter can only be used on update.</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RAW_COST_RATE_OVERRIDE</td>
<td>NUMBER</td>
<td>No</td>
<td>Override raw cost rate. This parameter can only be used on update.</td>
</tr>
<tr>
<td>BURD_COST_RATE_OVERRIDE</td>
<td>NUMBER</td>
<td>No</td>
<td>Override burdened cost rate. This parameter can only be used on update.</td>
</tr>
<tr>
<td>BILLABLE_WORK_PERCENT</td>
<td>NUMBER</td>
<td>No</td>
<td>Billable percent. This parameter can only be used on update.</td>
</tr>
<tr>
<td>MFG_COST_TYPE</td>
<td>VARCHAR2(10)</td>
<td>No</td>
<td>Manufacturing cost type. This parameter can only be used on update.</td>
</tr>
<tr>
<td>MFG_COST_TYPE_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the manufacturing cost type. This parameter can only be used on update.</td>
</tr>
<tr>
<td>P_CONTEXT_FLAG</td>
<td>VARCHAR2(1)</td>
<td>No</td>
<td>Flag indicating whether the task assignments that are not passed on the tasks should be deleted.</td>
</tr>
<tr>
<td>SCHEDULED_DELAY</td>
<td>NUMBER</td>
<td>No</td>
<td>The assignment scheduled delay</td>
</tr>
</tbody>
</table>

**ASSIGNMENT_OUT_REC_TYPE**

The following table shows the ASSIGNMENT_OUT_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Identifier of the task</td>
</tr>
<tr>
<td>PA_TASK_ASSIGNMENT_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Identifier of the task assignment</td>
</tr>
<tr>
<td>RESOURCE_ALIAS</td>
<td>VARCHAR2(80)</td>
<td>Yes</td>
<td>Alias of the planning resource</td>
</tr>
<tr>
<td>RESOURCE_LIST_MEMBER_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>Identifier of the planning resource</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>VARCHAR2(1)</td>
<td>Yes</td>
<td>Returned status</td>
</tr>
</tbody>
</table>
### TASK_DEL_REC_TYPE

The following table shows the TASK_DEL_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ELEM_VERSION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task version</td>
</tr>
<tr>
<td>PA_TASK_ASSIGNMENT_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task assignment</td>
</tr>
<tr>
<td>DEL_TA_FLAG</td>
<td>VARCHAR2(1)</td>
<td>No</td>
<td>Flag indicating whether the assignment should be deleted</td>
</tr>
</tbody>
</table>

### TASK_ASGMT_DEL_TYPE

The following table shows the TASK_ASGMT_DEL_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task</td>
</tr>
<tr>
<td>START_DEL_INDEX</td>
<td>NUMBER</td>
<td>No</td>
<td>Starting index</td>
</tr>
<tr>
<td>END_DEL_INDEX</td>
<td>NUMBER</td>
<td>No</td>
<td>Ending index</td>
</tr>
<tr>
<td>DEL_TA_FLAG</td>
<td>VARCHAR2(1)</td>
<td>No</td>
<td>Flag indicating whether to delete the assignment</td>
</tr>
</tbody>
</table>

### ASSIGNMENT_PERIODS_TYPE

The following table shows the ASSIGNMENTS_PERIODS_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_PRODUCT_CODE</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>Product code</td>
</tr>
<tr>
<td>PM_PROJECT_REFERENCE</td>
<td>VARCHAR2(25)</td>
<td>No</td>
<td>External project reference</td>
</tr>
<tr>
<td>PA_PROJECT_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the project</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>PA_STRUCTURE_VERSION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the structure version</td>
</tr>
<tr>
<td>PM_TASK_REFERENCE</td>
<td>VARCHAR2(25)</td>
<td>No</td>
<td>External task reference</td>
</tr>
<tr>
<td>PA_TASK_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the project task</td>
</tr>
<tr>
<td>PA_TASK_ELEMENT_VERSION_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task version</td>
</tr>
<tr>
<td>PM_TASK_ASGMT_REFERENCE</td>
<td>VARCHAR2(25)</td>
<td>No</td>
<td>External task assignment reference</td>
</tr>
<tr>
<td>PA_TASK_ASSIGNMENT_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the task assignment</td>
</tr>
<tr>
<td>RESOURCE_ALIAS</td>
<td>VARCHAR2(80)</td>
<td>No</td>
<td>Alias of the planning resource</td>
</tr>
<tr>
<td>RESOURCE_LIST_MEMBER_ID</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier of the planning resource</td>
</tr>
<tr>
<td>PERIOD_NAME</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>Name of the period</td>
</tr>
<tr>
<td>START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Starting date of the period</td>
</tr>
<tr>
<td>END_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Ending date of the period</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>NUMBER</td>
<td>No</td>
<td>Planned effort or quantity for the period</td>
</tr>
<tr>
<td>TXN_RAW_COST</td>
<td>NUMBER</td>
<td>No</td>
<td>Planned raw cost for the period</td>
</tr>
<tr>
<td>TXN_BURDENED_COST</td>
<td>NUMBER</td>
<td>No</td>
<td>Planned burdened cost for the period</td>
</tr>
<tr>
<td>TXN_CURRENCY_CODE</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>Currency code</td>
</tr>
</tbody>
</table>

**Task Assignment API Procedure Definitions**

This section contains description of the task assignments APIs, including business rules and parameters.
LOAD_TASK_ASSIGNMENTS

LOAD_TASK_ASSIGNMENTS is a PL/SQL procedure that moves task assignments from client side to a PLSQL table on the server side, where it is used by LOAD/EXECUTE/FETCH cycle.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- X_RETURN_STATUS
- X_MSG_COUNT
- X_MSG_DATA

LOAD_TASK_ASSIGNMENT_PERIODS

LOAD_TASK_ASSIGNMENT_PERIODS is a PL/SQL procedure that moves the periodic data of task assignments from client side to a PLSQL table on the server side.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- X_RETURN_STATUS
- X_MSG_COUNT
- X_MSG_DATA

EXECUTE_CREATE_TASK_ASGMTS

EXECUTE_CREATE_TASK_ASGMTS is a PL/SQL procedure that creates task assignments using data in a PLSQL table.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- X_RETURN_STATUS
- X_MSG_COUNT
• X_MSG_DATA

EXECUTE_UPDATE_TASK_ASGMTS
EXECUTE_UPDATE_TASK_ASGMTS is a PL/SQL procedure that updates task assignments.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for this procedure are listed below:
• X_RETURN_STATUS
• X_MSG_COUNT
• X_MSG_DATA

CREATE_TASK_ASSIGNMENTS
CREATE_TASK_ASSIGNMENTS is a PL/SQL procedure that creates task assignments by accepting a table of assignment records.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for this procedure are listed below:
• P_TASK_ASSIGNMENTS_IN
• P_TASK_ASSIGNMENTS_OUT
• X_RETURN_STATUS
• X_MSG_COUNT
• X_MSG_DATA

CREATE_TASK_ASSIGNMENT_PERIODS
CREATE_TASK_ASSIGNMENT_PERIODS is a PL/SQL procedure that creates task assignments and periodic data by accepting a table of assignment records and a table of periodic data.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for this procedure are listed below:

- P_TASK_ASSIGNMENTS_IN
- P_TASK_ASSIGNMENTS_OUT
- P_TASK_ASSIGNMENT_PERIODS_IN
- P_TASK_ASSIGNMENT_PERIODS_OUT
- X_RETURN_STATUS
- X_MSG_COUNT
- X_MSG_DATA

**UPDATE_TASK_ASSIGNMENTS**

UPDATE_TASK_ASSIGNMENTS is a PL/SQL procedure that updates task assignments. This API allows users to update the resource assignments on tasks without locking the workplan version, and allows multiple users to update the resource assignments for a task.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_TASK_ASSIGNMENTS_IN
- P_TASK_ASSIGNMENTS_OUT
- X_RETURN_STATUS
- X_MSG_COUNT
- X_MSG_DATA

**UPDATE_TASK_ASSIGNMENT_PERIODS**

UPDATE_TASK_ASSIGNMENT_PERIODS is a PL/SQL procedure that updates task assignments and periodic data.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_TASK_ASSIGNMENTS_IN
• P_TASK_ASSIGNMENTS_OUT
• P_TASK_ASSIGNMENT_PERIODS_IN
• P_TASK_ASSIGNMENT_PERIODS_OUT
• X_RETURN_STATUS
• X_MSG_COUNT
• X_MSG_DATA

DELETE_TASK_ASSIGNMENTS
DELETE_TASK_ASSIGNMENTS is a PL/SQL procedure that deletes task assignments.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for this procedure are listed below:
• P_TASK_ASSIGNMENTS_IN
• X_RETURN_STATUS
• X_MSG_COUNT
• X_MSG_DATA

FETCH_TASK_ASSIGNMENTS
FETCH_TASK_ASSIGNMENTS is a PL/SQL procedure that retrieves task assignments from the server side PLSQL table.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for this procedure are listed below:
• P_PM_TASK_ASGMT_REFERENCE
• P_PA_TASK_ASSIGNMENT_ID
• P_PM_TASK_REFERENCE
• P_PA_TASK_ID
• PRESOURCE_ALIAS
• PRESOURCE_LIST_MEMBER_ID
• X_RETURN_STATUS

CONVERT_PM_TAREF_TO_ID
CONVERT_PM_TAREF_TO_ID is a PL/SQL procedure that converts a given task assignment reference to a task assignment ID.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_PM_PRODUCT_CODE
• P_PA_PROJECT_ID
• P_PA_STRUCTURE_VERSION_ID
• P_PA_TASK_ID
• P_PA_TASK_ELEM_VER_ID
• X_PA_TASK_ASSIGNMENT_ID
• X_RETURN_STATUS

INIT_TASK_ASSIGNMENTS
INIT_TASK_ASSIGNMENTS initializes the task assignments global tables prior to Load-Execute cycle.

INIT_TASK_ASSIGNMENTS has no parameters.
This chapter describes how to implement APIs that interface and assign assets from external systems.

This chapter covers the following topics:

- Asset APIs
- Asset API Procedure and Function Definitions
- Cost Plus Application Programming Interface (API)

**Asset APIs**

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

**Asset API Views**

The following list shows 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.

- **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.
Asset API Procedures and Functions

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

- ADD_PROJECT_ASSET, page 4-2
- UPDATE_PROJECT_ASSET, page 4-3
- DELETE_PROJECT_ASSET, page 4-4
- ADD_ASSET_ASSIGNMENT, page 4-4
- DELETE_ASSET_ASSIGNMENT, page 4-4
- LOAD_PROJECT_ASSET, page 4-5
- LOAD_ASSET_ASSIGNMENT, page 4-5
- EXECUTE_ADD_PROJECT_ASSET, page 4-6
- CONVERT_PM_ASSTREF_TO_ID, page 4-7
- FETCH_PROJECT_ASSET_ID, page 4-7

Asset API Procedure and Function 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
• P_PA_PROJECT_ID
• P_PM_ASSET_REFERENCE
• P_PA_ASSET_NAME
• P_ASSET_NUMBER
• P_ASSET_DESCRIPTION
• P_PROJECT_ASSET_TYPE

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_MSG_COUNT
• P_MSG_DATA
• P_RETURN_STATUS
• P_PM_PRODUCT_CODE
• P_PM_PROJECT_REFERENCE
• P_PA_PROJECT_ID
• P_PA_ASSET_NAME
• P_ASSET_NUMBER
• P_ASSET_DESCRIPTION
• P_PROJECT_ASSET_TYPE
DELETE_PROJECT_ASSET

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameter for this procedure is listed below:

- P_API_VERSION_NUMBER

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_MSG_COUNT
- P_MSG_DATA
- P_RETURN_STATUS
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
- P_PA_PROJECT_ID
- P_PM_TASK_REFERENCE

DELETE_ASSET_ASSIGNMENT

This procedure deletes an asset assignment from a project.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_MSG_COUNT
• P_MSG_DATA
• P_RETURN_STATUS
• P_PM_PRODUCT_CODE
• P_PM_PROJECT_REFERENCE
• P_PA_PROJECT_ID
• P_PM_TASK_REFERENCE

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_PM_ASSET_REFERENCE
• P_PA_ASSET_NAME
• P_ASSET_NUMBER
• P_ASSET_DESCRIPTION
• P_PROJECT_ASSET_TYPE

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of
The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_RETURN_STATUS
- P_PM_TASK_REFERENCE

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_MSG_COUNT
- P_MSG_DATA
- P_RETURN_STATUS
- P_PM_PRODUCT_CODE
- P_PM_PROJECT_REFERENCE
- P_PA_PROJECT_ID
- P_ASSETS_IN
CONVERT_PM_ASSETREF_TO_ID

This procedure converts an incoming asset reference to a project asset ID.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_PA_PROJECT_ID
- P_RETURN_STATUS

FETCH_PROJECT_ASSET_ID

This function returns the PROJECT_ASSET_ID based on the ASSETREFERENCE and PROJECT_ID.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_PA_PROJECT_ID
- P_PM_ASSETREFERENCE

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.
Get Burden Amount

The cost plus application programming interface procedure is
PA_COST_PLUS.GET_BURDEN_AMOUNT.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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:

\[
\text{status} = 0 \quad \text{The procedure executed successfully.}
\]

\[
\text{status} < 0 \quad \text{An Oracle error occurred and the process did not complete.}
\]

**Tip:** 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.

\[
\text{status} > 0 \quad \text{See stage parameter.}
\]

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.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Cannot find a revision for the given burden schedule and effective date</td>
</tr>
<tr>
<td>200</td>
<td>Cannot find the burden structure</td>
</tr>
<tr>
<td>300</td>
<td>Expenditure type is not in a cost base in the burden structure</td>
</tr>
<tr>
<td>400</td>
<td>There is no active compiled set for the given burden schedule and organization</td>
</tr>
<tr>
<td>500</td>
<td>There is no compiled multiplier for the given qualification</td>
</tr>
</tbody>
</table>
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.

```plsql
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;
```
This chapter describes how to implement APIs for:

- Agreements and funding
- Events

This chapter covers the following topics:

- Agreement and Funding APIs
- Agreement and Funding API Procedure Definitions
- Using Agreement and Funding APIs
- Creating an Agreement Using Load-Execute-Fetch APIs
- Creating an Agreement Using a Composite Datatype API
- Event APIs
- Event API Procedure Definitions

**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.

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_AGREEMENT_TYPE_LOV_V</td>
<td>Retrieves valid agreement types</td>
</tr>
<tr>
<td>PA_TERMS_LOV_V</td>
<td>Retrieves customer terms</td>
</tr>
<tr>
<td>PA_OWNED_BY_LOV_V</td>
<td>Retrieves valid employees</td>
</tr>
<tr>
<td>PA_CUSTOMERS_LOV_V</td>
<td>Retrieves valid customer names and numbers</td>
</tr>
</tbody>
</table>

**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.

- CREATE_AGREEMENT, page 5-3
- DELETE_AGREEMENT, page 5-4
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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER

• P_PM_PRODUCT_CODE

• P_AGREEMENT_IN_REC

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
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 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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_MSG_COUNT
- P_MSG_DATA
- P_RETURN_STATUS
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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_MSG_COUNT
- P_MSG_DATA
- P_RETURN_STATUS
- P_PM_PRODUCT_CODE
• P_PA_PROJECT_ID

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER

• P_PM_PRODUCT_CODE

• P_PM_FUNDING_REFERENCE

• P_FUNDING_ID

• P_PA_PROJECT_ID

• P_AGREEMENT_ID
- P_ALLOCATED_AMOUNT

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PM_FUNDING_REFERENCE
- P_FUNDING_ID

UPDATE_FUNDING

This API updates funding 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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_PM_PRODUCT_CODE
• P_PM_FUNDING_REFERENCE
• P_FUNDING_ID
• P_AGREEMENT_ID

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_PM_AGREEMENT_REFERENCE
• P_AGREEMENT_ID
• P_CUSTOMER_ID
• P_CUSTOMER_NAME
• P_CUSTOMER_NUM
• P_AGREEMENT_NUM
• P_AGREEMENT_TYPE
• P_AMOUNT
• P_TERM_ID
• P_TERM_NAME
• P_OWNED_BY_PERSON_ID
• P_OWNED_BY_PERSON_NAME

LOAD_FUNDING

This API loads funding to a PL/SQL table.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_PM_FUNDING_REFERENCE
• P FUNDING_ID
• P AGREEMENT_ID
• P PROJECT_ID
• P_ALLOCATED_AMOUNT

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_AGREEMENT_ID_OUT
- P_CUSTOMER_ID_OUT

**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 than the revenue accrued or billed amount, the API does not allow the revenue or invoice limit to be changed.

- The agreement amount cannot 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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

**FETCH_FUNDING**

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_FUNDING_INDEX

**CLEAR_AGREEMENT**

This API clears the global variables that were set up during initialization.

**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 \(\geq\) total funding amount \(\geq 0\)
AND

total funding amount >= amount accrued or billed

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER

• P_PM_AGREEMENT_REFERENCE

• P_AGREEMENT_ID

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_PM_AGREEMENT_REFERENCE
• P_AGREEMENT_ID
• P_PM_FUNDING_REFERENCE
• P_TASK_ID
• P_PROJECT_ID

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 \geq total funding amount \geq 0
  AND
  total funding amount \geq amount accrued or billed

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_PM_FUNDING_REFERENCE
• P_FUNDING_ID

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_PM_PRODUCT_CODE
• P_PM_FUNDING_REFERENCE
• P_FUNDING_ID
• P_PM_AGREEMENT_REFERENCE
• P_AGREEMENT_ID

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 5-1

• 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 2-21.
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 3-60.

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 2-8.

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.

**Important:** 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**

After the Load procedures have successfully moved the agreement and funding data to the Oracle Projects global PL/SQL tables, call up the EXECUTE_CREATE_AGREEMENT procedure 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 2-25.

**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 5-24.
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_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.Agreement_Rec_In_Type;
  l_agreement_out_rec pa_agreement_pub.Agreement_Rec_Out_Type;

  --Oracle funding specific parameters
  l_funding_type pa_agreement_pub.funding_rec_in_type;
  l_funding_in_tbl pa_agreement_pub.funding_in_tbl_type;
  l_funding_out_tbl pa_agreement_pub.funding_out_tbl_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_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.funding_rec_in_type;
  l_funding_in pa_agreement_pub.funding_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;
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 5-19.
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-out NUMBER;
l_funding_id NUMBER;

--variables needed for Oracle Agreement specific parameters
l_pm_product_code VARCHAR2(25);
p_agreement_in_rec pa_agreement_pub.Agreement_Rec_In_type
p_agreement_out_rec pa_agreement_pub.Agreement_Rec_Out_type

--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 pa_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_type;

-- 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.project_funding_id;
  l_funding_in(i).agreement_id := l_funding_rec.agreement_id;
  l_funding_in(i).project_id := l_funding_rec.project_id;
  l_funding_in(i).task_id := l_funding_rec.task_id;
  l_funding_in(i).allocated_amount := l_funding_rec.allocated_amount;
  l_funding_in(i).date_allocated := l_funding_rec.date_allocated;
  l_funding_in(i).desc_flex_name := l_funding_rec.desc_flex_name;
  l_funding_in(i).attribute_category := l_funding_rec.attribute_category;
  l_funding_in(i).attribute1 := l_funding_rec.attribute1;
  l_funding_in(i).attribute2 := l_funding_rec.attribute2;
  l_funding_in(i).attribute3 := l_funding_rec.attribute3;
  l_funding_in(i).attribute4 := l_funding_rec.attribute4;
  l_funding_in(i).attribute5 := l_funding_rec.attribute5;
  l_funding_in(i).attribute6 := l_funding_rec.attribute6;
  l_funding_in(i).attribute7 := l_funding_rec.attribute7;
  l_funding_in(i).attribute8 := l_funding_rec.attribute8;
  l_funding_in(i).attribute9 := l_funding_rec.attribute9;
  l_funding_in(i).attribute10 := l_funding_rec.attribute10;
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
)
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.

- CREATE_EVENT, page 5-32
- DELETE_EVENT, page 5-32
- UPDATE_EVENT, page 5-32
- INIT_EVENT, page 5-33
- LOAD_EVENT, page 5-33
• EXECUTE_CREATE_EVENT, page 5-33
• EXECUTE_UPDATE_EVENT, page 5-34
• FETCH_EVENT, page 5-34
• CLEAR_EVENT, page 5-34
• CHECK_DELETE_EVENT_OK, page 5-34

Events API Record and Table Datatypes

The following PL/SQL record types and table types defined at the package specification level are used in the Events APIs. These PL/SQL record types and table types represent an array of data in the PA_EVENTS table.

EVENT_REC_IN_TYPE Datatype

The following table shows the parameters in EVENT_REC_IN_TYPE:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PM_EVENT_REFERENCE</td>
<td>VARCHAR2</td>
<td>NULL</td>
<td>Unique identifier of the event in the external system</td>
</tr>
<tr>
<td></td>
<td>(25)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_TASK_NUMBER</td>
<td>VARCHAR2</td>
<td>NULL</td>
<td>The sequential number that identifies the task</td>
</tr>
<tr>
<td></td>
<td>(25)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_EVENT_NUMBER</td>
<td>NUMBER</td>
<td>NULL</td>
<td>The sequential number that identifies the event</td>
</tr>
<tr>
<td>P_EVENT_TYPE</td>
<td>VARCHAR2</td>
<td>NULL</td>
<td>The event type that classifies the event</td>
</tr>
<tr>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_DESCRIPTION</td>
<td>VARCHAR2</td>
<td>NULL</td>
<td>The free text description of the event. For billing events, the description is displayed as the invoice line description.</td>
</tr>
<tr>
<td></td>
<td>(250)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_BILL_HOLD_FLAG</td>
<td>VARCHAR2</td>
<td>NULL</td>
<td>Flag indicating that the event is held from invoicing</td>
</tr>
<tr>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_COMPLETION_DATE</td>
<td>DATE</td>
<td>NULL</td>
<td>The date on which the event is complete, and on or after which the event is processed for revenue accrual and/or invoicing.</td>
</tr>
<tr>
<td>P_DESC_FLEX_NAME</td>
<td>VARCHAR2 (240)</td>
<td>NULL</td>
<td>Descriptive flexfield name</td>
</tr>
<tr>
<td>P_ATTRIBUTE_CATEGORY</td>
<td>VARCHAR2 (30)</td>
<td>NULL</td>
<td>Descriptive flexfield context field</td>
</tr>
<tr>
<td>P_ATTRIBUTE1 through ATTRIBUTE 10</td>
<td>VARCHAR2 (150)</td>
<td>NULL</td>
<td>Descriptive flexfield segment</td>
</tr>
<tr>
<td>P_PROJECT_NUMBER</td>
<td>VARCHAR2 (25)</td>
<td>NULL</td>
<td>The sequential number that identifies the project</td>
</tr>
<tr>
<td>P_ORGANIZATION_NAME</td>
<td>VARCHAR2 (240)</td>
<td>NULL</td>
<td>Name of the organization</td>
</tr>
<tr>
<td>P_INVENTORY_ORG_NAME</td>
<td>VARCHAR2 (240)</td>
<td>NULL</td>
<td>The inventory organization associated with the event</td>
</tr>
<tr>
<td>P_INVENTORY_ITEM_ID</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Identifier of the inventory item associated with the event</td>
</tr>
<tr>
<td>P_QUANTITY_BILLED</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Bill quantity</td>
</tr>
<tr>
<td>P_UOM_CODE</td>
<td>VARCHAR2 (3)</td>
<td>NULL</td>
<td>Unit of measure</td>
</tr>
<tr>
<td>P_UNIT_PRICE</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Contract price</td>
</tr>
<tr>
<td>P_REFERENCE1 THROUGH REFERENCE 10</td>
<td>VARCHAR2 (240)</td>
<td>NULL</td>
<td>Generic reference columns</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_BILL_TRANS_CURRENCY_CODE</td>
<td>VARCHAR2(15)</td>
<td>NULL</td>
<td>Transaction currency code of the event</td>
</tr>
<tr>
<td>P_BILL_TRANS_BILL_AMOUNT</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Bill amount in the event transaction</td>
</tr>
<tr>
<td>P_BILL_TRANS_REV_AMOUNT</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Revenue amount in the event transaction</td>
</tr>
<tr>
<td>P_PROJECT_RATE_TYPE</td>
<td>VARCHAR2(30)</td>
<td>NULL</td>
<td>Exchange rate type to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>project currency</td>
</tr>
<tr>
<td>P_PROJECT_RATE_DATE</td>
<td>DATE</td>
<td>NULL</td>
<td>Exchange rate date to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>project currency if Fixed Date rate date type is used for customer billing.</td>
</tr>
<tr>
<td>P_PROJECT_EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Exchange rate to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>project currency if User exchange rate type is used.</td>
</tr>
<tr>
<td>P_PROJFUNC_RATE_TYPE</td>
<td>VARCHAR2(30)</td>
<td>NULL</td>
<td>Exchange rate type to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>project functional currency for customer billing.</td>
</tr>
<tr>
<td>P_PROJFUNC_RATE_DATE</td>
<td>DATE</td>
<td>NULL</td>
<td>Exchange rate date to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>project functional currency if Fixed Date rate date type is used for customer billing.</td>
</tr>
<tr>
<td>P_PROJFUNC_EXCHANGE_RATE</td>
<td>NUMBER</td>
<td>NULL</td>
<td>Exchange rate to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>project functional if User exchange rate type is used.</td>
</tr>
<tr>
<td>P_FUNDING_RATE_TYPE</td>
<td>VARCHAR2(30)</td>
<td>NULL</td>
<td>Exchange rate type to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>funding currency for customer billing.</td>
</tr>
<tr>
<td>P_FUNDING_RATE_DATE</td>
<td>DATE</td>
<td>NULL</td>
<td>Exchange rate date to use for conversion from bill transaction currency to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>funding currency if Fixed Date rate date type is used for customer billing.</td>
</tr>
</tbody>
</table>

---

5-30  Oracle Projects APIs, Client Extensions, and Open Interfaces Reference
### Oracle Project Billing APIs

#### P_FUNDING_EXCHANGE_RATE
- **Type:** NUMBER
- **Value:** NULL
- **Description:** Exchange rate to use for conversion from bill transaction currency to project or functional currency

#### P_ADJUSTING_REVENUE_FLAG
- **Type:** VARCHAR2(n=1)
- **Value:** NULL
- **Description:** Flag indicating that the event is an adjusting revenue event

#### P_EVENT_ID
- **Type:** NUMBER
- **Value:** NULL
- **Description:** Identifier of the primary key for table

#### P_DELIVERABLE_ID
- **Type:** NUMBER
- **Value:** NULL
- **Description:** Identifier of the deliverable

#### P_ACTION_ID
- **Type:** NUMBER
- **Value:** NULL
- **Description:** Identifier of the action

#### P_CONTEXT
- **Type:** VARCHAR2(n=1)
- **Value:** NULL
- **Description:** Currently not used (for future development)

#### P_RECORD_VERSION_NUMBER
- **Type:** NUMBER
- **Value:** NULL
- **Description:** The login number

---

**EVENT_REC_OUT_TYPE Datatype**

The following table displays the parameters for EVENT_REC_OUT_TYPE:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PM_EVENT_REFERENCE</td>
<td>VARCHAR(25)</td>
<td>NULL</td>
<td>Unique identifier of an event in external system</td>
</tr>
<tr>
<td>EVENT_ID</td>
<td>NUMBER (15)</td>
<td>NULL</td>
<td>The reference code that uniquely identifies the event in Oracle Projects</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>VARCHAR(1)</td>
<td>NULL</td>
<td>API standard</td>
</tr>
</tbody>
</table>

**EVENT_IN_TBL_TYPE**

EVENT_IN_TBL_TYPE is the table of EVENT_REC_IN_TYPE.
**EVENT_OUT_TBL_TYPE**

EVENT_OUT_TBL_TYPE is the table of EVENT_REC_OUT_TYPE.

---

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

**DELETE_EVENT**

This API deletes an event.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_EVENT_REFERENCE
- P_EVENT_ID

**UPDATE_EVENT**

This API updates an event or set of events.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
• P_MSG_COUNT
• P_MSG_DATA
• P_RETURN_STATUS

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_PM_PRODUCT_CODE
• P_API_VERSION_NUMBER
• P_PM_EVENT_REFERENCE
• P_TASK_NUMBER
• P_EVENT_NUMBER
• P_EVENT_TYPE

EXECUTE_CREATE_EVENT

This API creates an event using the data which is stored in the global tables during the Load phase.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:
• P_API_VERSION_NUMBER
• P_PM_PRODUCT_CODE
• P_EVENT_ID_OUT
**EXECUTE_UPDATE_EVENT**

This API updates event data using the information stored in the global tables during the Load phase.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameter for this procedure is listed below:

- P_API_VERSION_NUMBER

**FETCH_EVENT**

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

**CLEAR_EVENT**

This API clears the global variables that were set up during initialization.

**Parameters:** None

**CHECK_DELETE_EVENT_OK**

This API checks whether an event can be deleted.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PM_EVENT_REFERENCE
• P_EVENT_ID
This chapter describes how to implement Oracle Project Management APIs.

This chapter covers the following topics:

- Project Deliverables APIs
- Budget APIs
- Budget Record and Table Datatypes
- Budget API Procedure Definitions
- Using Budget APIs
- Creating a Budget Using the Load-Execute-Fetch APIs
- Creating a Budget Using a Composite Datatype API
- Refresh Planning Amounts API
- Status APIs
- Record and Table Datatypes
- Required Parameters and Parameter Values
- Status API Procedure Definitions
- Custom Summarization Reporting APIs
- Project Performance Reporting APIs

**Project Deliverables APIs**

The APIs for project deliverables provide an interface for external systems to create, update, and delete deliverables for a project. They also include APIs that enable you to create or delete deliverable associations to tasks and task assignments.
Project Deliverables API Views

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

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_DELIVERABLES_AMG_V</td>
<td>The list of deliverables for a project</td>
</tr>
<tr>
<td>PA_DLVR_ACTIONS_AMG_V</td>
<td>The list of deliverable actions for a project</td>
</tr>
<tr>
<td>PA_DELIVERABLE_TYPES_AMG_V</td>
<td>The list of deliverable types</td>
</tr>
<tr>
<td>PA_DELIVERABLE_STATUSES_AMG_V</td>
<td>The list of valid deliverable statuses</td>
</tr>
<tr>
<td>PA_ACTION_FUNCTIONS_AMG_V</td>
<td>The list of action functions</td>
</tr>
</tbody>
</table>

Project Deliverables API Procedures

The API procedures provided for project deliverables are listed below. These procedures are located in the public API package PA_PROJECT_PUB.

- LOAD_DELIVERABLE, page 6-3
- LOAD_DELIVERABLES, page 6-3
- LOAD_ACTION, page 6-3
- LOAD_ACTIONS, page 6-4
- CREATE_DELIVERABLE, page 6-4
- CREATE_DELIVERABLE_ACTION, page 6-5
- UPDATE_DELIVERABLE, page 6-4
- UPDATE_DELIVERABLE_ACTION, page 6-5
- DELETE_DELIVERABLES, page 6-6
- DELETE_DELIVERABLE_ACTIONS, page 6-6
- ASSOCIATE_DLVR_TO_TASK, page 6-6
• ASSOCIATE_DLV_TO_TASK_ASSIGN, page 6-6
• DELETE_DLV_TO_TASK_ASSCN, page 6-7
• DELETE_DLV_TO_TASK_ASSIGN, page 6-7

You can view descriptions of all of the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

LOAD_DELIVERABLE
You can use this API to load the PL/SQL table for one deliverable. The required parameters for this procedure are listed below:
• X_RETURN_STATUS
• P_DELIVERABLE_NAME
• P_DELIVERABLE_SHORT_NAME
• PX_DELIVERABLE_ID
• P_PM_DELIVERABLE_REFERENCE

LOAD_DELIVERABLES
You can use this API to load the PL/SQL table for deliverables. The required parameters for this procedure are listed below:
• X_RETURN_STATUS
• P_DELIVERABLE_NAME
• P_DELIVERABLE_SHORT_NAME
• PX_DELIVERABLE_ID
• P_PM_DELIVERABLE_REFERENCE

LOAD_ACTION
You can use this procedure to load a single deliverable action in the PL/SQL table for deliverable actions. The required parameters for this procedure are listed below:
• P_DELIVERABLE_ID
LOAD_ACTIONS

You can use this procedure to load the PL/SQL table for deliverable actions. The required parameters for this procedure are listed below:

- **P_DELIVERABLE_ID**

Business Rules

- The parameters **P_EXPENDITURE_ORG_ID**, **P_EXPENDITURE_TYPE**, and **P_EXPENDITURE_ITEM_DATE** are required only if the destination type is **EXPENSE**.

- The parameter **P_RECEIVING_ORG_ID** is required only for non-item-based deliverables.

- A value is required for the parameter **P_RECEIVING_ORG** only for deliverables that are not of the type class **item**.

- The parameters **P_QUANTITY** and **P_UOM_CODE** are required only for non-item deliverable procurement type. These parameters are applicable for the action **Shipping**.

- The parameter **P_SHIP_FROM_ORGANIZATION_ID** is required only for Shipping actions for deliverables that are not of the type class **item**.

CREATE_DELIVERABLE

You can use this procedure to create a deliverable for a project. The required parameters for this procedure are listed below:

- **X_RETURN_STATUS**

- **P_DELIVERABLE_NAME**

- **P_DELIVERABLE_SHORT_NAME**

- **PX_DELIVERABLE_ID**

- **P_PM_DELIVERABLE_REFERENCE**

UPDATE_DELIVERABLE

You can use this procedure to update the attributes of a deliverable. The required parameters for this procedure are listed below:

- **X_RETURN_STATUS**
CREATE_DELIVERABLE_ACTION

You can use this procedure to create a deliverable action for a deliverable. The required parameters for this procedure are listed below:

- P_DELIVERABLE_ID

Business Rules

- If the class type of the deliverable is item, the following rules apply:
  - If the action is Procurement, values are required for P_RECEIVING_ORG_ID and P_QUANTITY.
  - If the action is Shipping, values are required for P_SHIP_FROM_ORGANIZATION_ID.

- If the destination type is EXPENSE, the following parameters are required:
  - P_EXPENDITURE_TYP
  - P_EXPENDITURE_ITEM_DATE

- If the deliverable class type is not item, the following rules apply:
  - The parameter P_RECEIVING_ORG_ID is required.
  - If the action is Procurement or Shipping, P_QUANTITY and P_UOM are required.
  - If the action is Shipping, P_SHIP_FROM_ORGANIZATION_ID is required.

UPDATE_DELIVERABLE_ACTION

You can use this procedure to update attributes of a deliverable action. The required parameters for this procedure are listed below:

- P_DELIVERABLE_ID

Business Rules

- If the class type of the deliverable is item, the following rules apply:
  - If the action is Procurement, values are required for P_RECEIVING_ORG_ID
• If the action is Shipping, values are required for P_SHIP_FROM_ORGANIZATION_ID.

• If the action is Procurement or Shipping, P_QUANTITY and P_UOM are required.

• If the action is Shipping, P_SHIP_FROM_ORGANIZATION_ID is required.

DELETE_DELIVERABLES
You can use this procedure to delete a deliverable for a project.

DELETE_DELIVERABLE_ACTIONS
You can use this procedure to delete a deliverable action for a deliverable. The required parameters for this procedure are listed below:

• P_PM_DELIVERABLE_REFERENCE

• P_DELIVERABLE_ID

• P_ACTION_ID

ASSOCIATE_DLV_TO_TASK
You can use this procedure to associate a deliverable with a task. The required parameters for this procedure are listed below:

• P_DELIVERABLE_REFERENCE

• P_DELIVERABLE_ID

ASSOCIATE_DLV_TO_TASK_ASSIGN
You can use this procedure to associate a deliverable to a task assignment. The required parameters for this procedure are listed below:
• P_DELIVERABLE_ID

**DELETE_DLV_TO_TASK_ASSCN**

You can use this procedure to delete a deliverable-to-task association. The required parameters for this procedure are listed below:

• P_PM_DELIVERABLE_REFERENCE

• P_DELIVERABLE_ID

**DELETE_DLV_TO_TASK_ASSIGN**

You can use this procedure to delete a deliverable-to-task assignment association. The required parameters for this procedure are listed below:

• P_DELIVERABLE_ID

• P_PM_DELIVERABLE_REFERENCE

**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 PRESOURCE_LIST_NAME or the P_RESOURCE_LIST_ID parameter to identify the resource list.

**Budget API Views**

The following list shows 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.
<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_BASE_BUDGET_BY_GL_PERIOD_V</td>
<td>Most recent baselined budget amounts by GL period</td>
</tr>
<tr>
<td>PA_BASE_BUDGET_BY_PA_PERIOD_V</td>
<td>Most recent baselined budget amounts by PA period</td>
</tr>
<tr>
<td>PA_BUDGET_CHANGE_REASON_V</td>
<td>Retrieves budget change reason codes</td>
</tr>
<tr>
<td>PA_BUDGET_ENTRY_METHODS_V</td>
<td>Retrieves budget entry methods</td>
</tr>
<tr>
<td>PA_BUDGET_STATUS_CODES_V</td>
<td>Retrieves budget status codes</td>
</tr>
<tr>
<td>PA_BUDGET_TYPES_V</td>
<td>Retrieves budget types</td>
</tr>
<tr>
<td>PA_ORIG_BUDGET_BY_GL_PERIOD_V</td>
<td>Original budget amounts by GL period.</td>
</tr>
<tr>
<td>PA_ORIG_BUDGET_BY_PA_PERIOD_V</td>
<td>Original budget amounts by PA period</td>
</tr>
<tr>
<td>PA_FINPLAN_TYPES_V</td>
<td>Retrieves financial plan types</td>
</tr>
<tr>
<td>PA_BASE_FINPLAN_BY_GL_PERIOD_V</td>
<td>Most recent budget baseline amounts by GL period for financial plan types</td>
</tr>
<tr>
<td>PA_BASE_FINPLAN_BY_PA_PERIOD_V</td>
<td>Most recent budget baseline amounts by PA period for financial plan types</td>
</tr>
<tr>
<td>PA_ORIG_FINPLAN_BY_GL_PERIOD_V</td>
<td>Original budget amounts by GL period for financial plan types</td>
</tr>
<tr>
<td>PA_ORIG_FINPLAN_BY_PA_PERIOD_V</td>
<td>Original budget amounts by PA period for financial plan types</td>
</tr>
</tbody>
</table>

**Budget API Procedures**

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

- Budget and Budget Line Procedures
  - ADD_BUDGET_LINE, page 6-16
  - BASELINE_BUDGET, page 6-20
• CALCULATE_AMOUNTS, page 6-21
• CREATE_DRAFT_BUDGET, page 6-24
• CREATE_DRAFT_FINPLAN, page 6-27
• DELETE_BUDGET_LINE, page 6-33
• DELETE_DRAFT_BUDGET, page 6-37
• GET_PROJECT_ID, page 6-39
• SET_PROJECT_ID, page 6-39
• UPDATE_BUDGET, page 6-39
• UPDATE_BUDGET_LINE, page 6-47

• Load-Execute-Fetch Procedures
  • CLEAR_BUDGET, page 6-52
  • CLEAR_CALCULATE_AMOUNTS, page 6-52
  • EXECUTE_CALCULATE_AMOUNTS, page 6-52
  • EXECUTE_CREATE_DRAFT_BUDGET, page 6-53
  • EXECUTE_CREATE_DRAFT_FINPLAN, page 6-54
  • EXECUTE_UPDATE_BUDGET, page 6-54
  • FETCH_BUDGET_LINE, page 6-55
  • FETCH_CALCULATE_AMOUNTS, page 6-55
  • INIT_BUDGET, page 6-55
  • INIT_CALCULATE_AMOUNTS, page 6-55
  • LOAD_BUDGET_LINE, page 6-55
  • LOAD_RESOURCE_INFO, page 6-56

Budget Record and Table Datatypes
The record and table datatypes used in the APIs are defined on the following pages.
**BUDGET_LINE_IN_TBL_TYPE Datatype**

The table type BUDGET_LINE_IN_TBL_TYPE is a table of BUDGET_LINE_IN_REC_TYPE. The following table shows the attributes of the BUDGET_LINE_IN_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The reference code that uniquely identifies the task in Oracle Projects</td>
</tr>
<tr>
<td>PM_TASKREFERENCE</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The reference code that uniquely identifies the task in the external system</td>
</tr>
<tr>
<td>RESOURCE_ALIAS</td>
<td>IN</td>
<td>VARCHAR2(80)</td>
<td>No</td>
<td>The alias of a resource</td>
</tr>
<tr>
<td>RESOURCE_LIST</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The identification code of the resource</td>
</tr>
<tr>
<td>BUDGET_START_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>Start date of budget line</td>
</tr>
<tr>
<td>BUDGET_END_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>End date of budget line</td>
</tr>
<tr>
<td>PERIOD_NAME</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>GL or PA period name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>IN</td>
<td>VARCHAR2(255)</td>
<td>No</td>
<td>(currently unavailable)</td>
</tr>
<tr>
<td>RAW_COST</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted raw cost amount</td>
</tr>
<tr>
<td>BURDENED_COST</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted burdened cost amount</td>
</tr>
<tr>
<td>REVENUE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted revenue amount</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
<td>---------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted quantity</td>
</tr>
<tr>
<td>PM_PRODUCT_CODE</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The product code of the vendor of the external system</td>
</tr>
<tr>
<td>PM_BUDGET_LINE_REFERENCE</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The reference code that identifies the budget line on client side</td>
</tr>
<tr>
<td>ATTRIBUTE_CATEGORY</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>Used by descriptive flexfields</td>
</tr>
<tr>
<td>ATTRIBUTE1 through ATTRIBUTE15</td>
<td>IN</td>
<td>VARCHAR2(150)</td>
<td>No</td>
<td>Budget line descriptive flexfield</td>
</tr>
<tr>
<td>TXN_CURRENCY_CODE</td>
<td>IN</td>
<td>VARCHAR2(15)</td>
<td>Yes</td>
<td>The transaction currency code for the budget line</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The rate type for converting cost amounts from the transaction currency to the project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_DATE_TEPE</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The rate date for converting cost amounts from transaction currency to project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>The rate date for converting cost amounts from transaction currency to project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The rate for converting cost amounts from the transaction currency to the project functional currency when the PROJFUNC_COST_RATE_TYPE is User</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROJFUNC_REV_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The rate type for converting revenue amounts from the transaction currency to the project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_REV_RATE_DATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The rate date type for converting revenue amounts from transaction currency to project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_REV_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>The rate date for converting revenue amounts from transaction currency to project functional currency</td>
</tr>
<tr>
<td>PROJFUNC_REV_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The rate for converting revenue amounts from the transaction currency to the project functional currency when the PROJFUNC_REV_RATE_TYPE is User</td>
</tr>
<tr>
<td>PROJECT_COST_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The rate type for converting cost amounts from the transaction currency to the project currency</td>
</tr>
<tr>
<td>PROJECT_COST_RATE_DATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The rate date type for converting cost amounts from transaction currency to project currency</td>
</tr>
<tr>
<td>PROJECT_COST_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>The rate date for converting cost amounts from transaction currency to project currency</td>
</tr>
<tr>
<td>PROJECT_COST_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The rate for converting cost amounts from the transaction currency to the project currency when the PROJECT_COST_RATE_TYPE is User</td>
</tr>
<tr>
<td>PROJECT_REV_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The rate type for converting revenue amounts from the transaction currency to the project currency</td>
</tr>
<tr>
<td>PROJECT_REV_RATE_DATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The rate date type for converting revenue amounts from transaction currency to project currency</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------</td>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PROJECT_REV_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>The rate date for converting revenue amounts from transaction currency to project currency</td>
</tr>
<tr>
<td>PROJECT_REV_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The rate for converting revenue amounts from the transaction currency to the project currency when the PROJECT_REV_RATE_TYPE is User</td>
</tr>
<tr>
<td>CHANGE_REASON_CODE</td>
<td>IN</td>
<td>VARCHAR2(30)</td>
<td>No</td>
<td>The reference code that identifies the change reason</td>
</tr>
</tbody>
</table>

**BUDGET_LINE_OUT_TBL_TYPE Datatype**

The table type BUDGET_LINE_OUT_TBL_TYPE is a table of BUDGET_LINE_OUT_REC_TYPE. The following table shows the attributes of the BUDGET_LINE_OUT_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN_STATUS</td>
<td>OUT NOCOPY</td>
<td>VARCHAR2(1)</td>
<td>Return status</td>
</tr>
</tbody>
</table>

**CALC_BUDGET_LINE_OUT_TBL_TYPE Datatype**

The table type CALC_BUDGET_LINE_OUT_TBL_TYPE is a table of CALC_BUDGET_LINE_OUT_REC_TYPE. The following table shows the attributes of the CALC_BUDGET_LINE_OUT_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ID</td>
<td>OUT NOCOPY</td>
<td>NUMBER</td>
<td>The reference code that uniquely identifies the task in Oracle Projects</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PM_TASK_Reference</td>
<td>OUT</td>
<td>VARCHAR2 (30)</td>
<td>The reference code that uniquely identifies the task in the external system</td>
</tr>
<tr>
<td>Resource_ALIASES</td>
<td>OUT</td>
<td>VARCHAR2 (80)</td>
<td>Alias of a resource</td>
</tr>
<tr>
<td>Resource_LIST_MEMBER_ID</td>
<td>NOCOPY</td>
<td>NUMBER</td>
<td>The identification code of the resource</td>
</tr>
<tr>
<td>Budget_Start_DATE</td>
<td>OUT</td>
<td>DATE</td>
<td>Start date of a budget</td>
</tr>
<tr>
<td>Budget_End_DATE</td>
<td>OUT</td>
<td>DATE</td>
<td>End date of a budget</td>
</tr>
<tr>
<td>Period_Name</td>
<td>OUT</td>
<td>VARCHAR2 (30)</td>
<td>PA or GL period name</td>
</tr>
<tr>
<td>Calculated_Raw_Cost</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated raw cost in transaction currency (P_TXN_CURRENCY_CODE)</td>
</tr>
<tr>
<td>Calculated_Burdened_Cost</td>
<td>NOCOPY</td>
<td>NUMBER</td>
<td>Calculated burdened cost in transaction currency (P_TXN_CURRENCY_CODE)</td>
</tr>
<tr>
<td>Calculated_Revenue</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated revenue in transaction currency (P_TXN_CURRENCY_CODE)</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Quantity</td>
</tr>
<tr>
<td>RETURN_STATUS</td>
<td>OUT</td>
<td>VARCHAR R2(1)</td>
<td>API standard</td>
</tr>
<tr>
<td>TXN_CURRENCY_CODE</td>
<td>OUT</td>
<td>VARCHAR R2(30)</td>
<td>The transaction currency code for the budget line. For Forms-based budgets, this is always the project functional currency.</td>
</tr>
<tr>
<td>PROJECT_RAW_COST</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated raw cost amount in project currency. Applicable only for budgets created using the web-based user interface.</td>
</tr>
<tr>
<td>PROJECT_BURDENED_COST</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated burdened cost amount in project currency. Applicable only for budgets created using the web-based user interface.</td>
</tr>
<tr>
<td>PROJECT_REVENUE</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated revenue amount in project currency. Applicable only for budgets created using the web-based user interface.</td>
</tr>
<tr>
<td>PROJFUNC_RAW_COST</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated raw cost amount in project functional currency. Applicable only for budgets created using the web-based user interface.</td>
</tr>
<tr>
<td>PROJFUNC_BURDENED_COST</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated burdened cost in project functional currency. Applicable only for budgets created using the web-based user interface.</td>
</tr>
<tr>
<td>PROJFUNC_REVENUE</td>
<td>OUT</td>
<td>NUMBER</td>
<td>Calculated revenue amount in project functional currency. Applicable only for budgets created using the web-based user interface.</td>
</tr>
</tbody>
</table>

**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 line to a draft or working version in Oracle Projects for either a project and budget type, or a project and financial plan type.

Business Rules

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

General Rules

- After you use ADD_BUDGET_LINE to add a line to a draft or working version, save the data before you call the procedure BASELINE_BUDGET. (A working version may require approval before you can create a baseline.) Enter funding in Oracle Projects before you create a baseline for a revenue budget.

- You can add lines only to draft or working versions. You cannot add lines to a submitted or baseline version. You cannot add lines to an approved revenue budget working version if the autobaseline feature is enabled.

- Oracle Projects establishes the following links between information stored in your system and certain information in Oracle Projects. You can use the following parameters instead of their corresponding Oracle Projects identification codes:
  - P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
  - P_FIN_PLAN_TYPE_NAME links to P_FIN_PLAN_TYPE_ID.
  - 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.

- You do not need to specify values for the following pair of parameters if the version is not categorized by resources:
  - PRESOURCE_ALIAS and PRESOURCE_LIST_MEMBER_ID

- The following rules apply for versions that are time-phased by PA or GL periods:
  - If a valid value is specified for the parameter P_PERIOD_NAME, then this procedure adds a line to the version for the specified period.
If a version is time-phased by PA periods, then the value specified for the parameter P_PERIOD_NAME must map to a PA period. Likewise, if a version is time-phased by GL periods, then the value specified for P_PERIOD_NAME must map to a GL period.

If the value specified for P_PERIOD_NAME is invalid, then the procedure will abort.

- If values are specified for the parameter P_BUDGET_START_DATE and the parameter P_PERIOD_NAME, then Oracle Projects uses P_PERIOD_NAME to determine the period.

- If no value is specified for the parameter P_PERIOD_NAME, then the procedures uses the parameter P_BUDGET_START_DATE to determine the period.

  If a version is time-phased by PA periods, then the procedure selects the period based on the PA calendar. If a version is time-phased by GL periods, then the procedure selects the period based on the GL calendar.

  If Oracle Projects cannot determine a valid period name, then the procedure will abort.

- If no values are specified for the parameter P_BUDGET_START_DATE and the parameter P_PERIOD_NAME, then the procedure will abort.

- If Descriptive Flexfields are defined, then you can pass them as IN parameters.

- All business rules for adding a line to a version for a budget type or a financial plan type in the user interface are applicable when you use this procedure.

**Budget Types**

- The following parameters are not used for planning with budget types:
  
  - P_FIN_PLAN_TYPE_ID
  
  - P_FIN_PLAN_TYPE_NAME
  
  - P_VERSION_NUMBER
  
  - P_VERSION_TYPE
  
  - P_CURRENCY_CODE

- You cannot use this procedure to add a budget line for a forecast budget type.

- Specify values for the parameters P_PA_TASK_ID or P_PM_TASK_REFERENCE only when planning by tasks, as defined by the budget entry method.
• Specify values for the parameter P_PERIOD_NAME only when planning by PA or GL period, as defined by the budget entry method.

• The task level at which you specify 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 procedure should pass only lowest tasks.

• When the budget entry method (BEM) restricts entry of a certain field as shown in the following table, do not specify values for the corresponding parameter.

<table>
<thead>
<tr>
<th>BEM Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost quantity not enterable</td>
<td>p_quantity</td>
</tr>
<tr>
<td>Raw cost not enterable</td>
<td>p_raw_cost</td>
</tr>
<tr>
<td>Burdened cost not enterable</td>
<td>p_burdened_cost</td>
</tr>
<tr>
<td>Revenue quantity not enterable</td>
<td>p_quantity</td>
</tr>
<tr>
<td>Revenue not enterable</td>
<td>p_revenue</td>
</tr>
</tbody>
</table>

• If a project and a budget type are enabled for budgetary controls, then you cannot add a budget line for a GL period that is after the latest encumbrance year defined in the ledger associated with the project.

**Financial Plan Types**

• The following parameter is not used for planning with financial plan types:
  • P_BUDGET_TYPE_CODE

• If no value is specified for the parameter P_VERSION_NUMBER, then this procedure adds the line to the current working version.

• You can specify values for the parameters P_PA_TASK_ID or P_PM_TASK_REFERENCE only when planning by tasks, as defined by the planning level of the budget or forecast.

• The task level at which you specify plan information should correspond to the level specified in the planning level of a budget or forecast version. For example, if the planning level specifies that you can enter a budget or forecast at the top task level, then this procedure should pass only top task and project-level budget and forecast.
amounts.

- You cannot use this procedure to add a line under the following conditions:
  - If the option to allow edit after initial baseline is not enabled for the financial plan type, then you cannot modify a working version after you create a baseline.
  - A budget or forecast version is locked by another user or is locked for processing
  - A budget or forecast version has processing errors
  - A change document version
  - A version for an organization forecast
  - A workplan version

- When the amount entry options restrict entry of certain fields as shown in the following table, do not specify values for the corresponding parameters. In these cases, the budget or forecast is created with the specified amount entry options and edit of the corresponding quantities and amounts is not allowed.

<table>
<thead>
<tr>
<th>Amount Entry Option Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost quantity not enterable</td>
<td>p_quantity (used when the version is for cost only)</td>
</tr>
<tr>
<td>Revenue quantity not enterable</td>
<td>p_quantity (used when the version is for revenue only)</td>
</tr>
<tr>
<td>Quantity not enterable</td>
<td>p_quantity (used when cost and revenue are planned together in the same version)</td>
</tr>
<tr>
<td>Raw cost not enterable</td>
<td>p_raw_cost</td>
</tr>
<tr>
<td>Burdened cost not enterable</td>
<td>p_burdened_cost</td>
</tr>
<tr>
<td>Revenue not enterable</td>
<td>p_revenue</td>
</tr>
</tbody>
</table>

- You can specify values for the parameters P_RAW_COST and P_BURDENED_COST only for budgets or forecasts for which the financial plan type allows planning for cost, or for cost and revenue together in the same version.
• You can specify a value for the parameter `P_REVENUE_AMOUNT` only for a budget or forecast for which the financial plan type allows planning for revenue, or for cost and revenue together in the same version.

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

• `P_API_VERSION_NUMBER`

• `P_PM_PRODUCT_CODE`

BASELINE_BUDGET

BASELINE_BUDGET is a PL/SQL procedure used to create a baseline for an existing budget in Oracle Projects for either a project and budget type, or a project and financial plan type.

Business Rules

• The following parameters are used for planning with financial plan types. These parameters are not used for planning with budget types.
  
  • `P_FIN_PLAN_TYPE_ID`
  
  • `P_FIN_PLAN_TYPE_NAME`
  
  • `P_VERSION_TYPE`

• You must set up funding in Oracle Projects before you can create a baseline for a revenue budget.

• If you have not yet submitted a budget, Oracle Projects submits it automatically before creating a baseline.

• 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 create a baseline for the first time, the `P_MARK_AS_ORIGINAL` is set to Y.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

• `P_API_VERSION_NUMBER`
• P_PM_PRODUCT_CODE
• P_BUDGET_TYPE_CODE

CALCULATE_AMOUNTS

CALCULATE_AMOUNTS is a PL/SQL procedure used to recalculate raw cost, burdened cost, and revenue amounts for lines in a draft or working version in Oracle Projects for either a project and a budget type, or a project and financial plan type. This procedure uses the Budget Calculation Extension (PA_CLIENT_EXTN_BUDGET) to perform the calculations.

Business Rules

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

General Rules

• You can recalculate amounts only for draft or working versions. You cannot recalculate amounts for submitted or baseline versions. You cannot use this procedure to recalculate amounts for an approved revenue budget version if the autobaseline feature is enabled.

• To call this procedure, you must specify the project identifier (either P_PA_PROJECT_ID or P_PM_PROJECT_REFERENCE), and one of the following budget/forecast identifiers:
  • P_BUDGET_VERSION_ID
  • BUDGET_TYPE_CODE
  • P_FIN_PLAN_TYPE_ID or P_FIN_PLAN_TYPE_NAME

• Because this procedure calls the Budget Calculation Client Extension (PA_CLIENT_EXTN_BUDGET), you must modify the extension to calculate the amounts you want.

After this procedure calls the extension procedure CALC_RAW_COST, Oracle Projects sets quantity and rate amounts as follows:

• For non-rate-based planning transactions, the quantity is set to the returned raw cost value. The raw cost rate remains unchanged (value equals 1), and the burdened cost rate is set to the burdened cost divided by the derived quantity.

• For rate-based planning transactions, the raw cost rate is set to the returned raw cost value divided by the quantity.
After this procedure calls the extension procedure CALC_BURDENED_COST, Oracle Projects sets the burden cost rate to the returned burdened cost divided by the quantity.

After this procedure calls the extension procedure CALC_REVENUE, Oracle Projects sets quantity and rate amounts as follows:

- For non-rate-based planning transactions from revenue-only plans, the quantity is set to the returned revenue value.

- For non-rate based planning transactions from revenue and cost plans, the bill rate is set to the returned revenue value divided by the quantity.

- For rate-based planning transactions, the bill rate is set to the returned revenue value divided by the quantity.

- To specify amounts to recalculate, set the value of one or more of the following parameters to Y:
  - P_CALC_RAW_COST_YN
  - P_CALC_BURDENED_COST_YN
  - P_CALC_REVENUE_YN

- To automatically update version lines with recalculated amounts after the successful execution of CALCULATE_AMOUNTS, set the parameter P_UPDATE_DB_FLAG to Y.

Regardless of the update status, CALCULATE_AMOUNTS returns one row of amounts for each line it reads. The procedure updates project currency and project functional currency amounts. Updated amounts are rolled up to the resource assignment level and version level when summarization processes are run.

Budget Types

- The following parameters are not used for planning with budget types:
  - P_FIN_PLAN_TYPE_ID
  - P_FIN_PLAN_TYPE_NAME
  - P_VERSION_TYPE
  - P_BUDGET_VERSION_NUMBER

- If a project and a budget type are enabled for budgetary controls, and the value for the parameter P_UPDATE_DB_FLAG is set to Y, then Oracle Projects does not recalculate budget lines for GL periods that are after the latest encumbrance year.
Financial Plan Types

- The following parameter is not used for planning with financial plan types:
  - `P_BUDGET_TYPE_CODE`

- If no value is specified for the parameter `P_BUDGET_VERSION_NUMBER`, then this procedure recalculates amounts for the current working version.

- You must specify a value for the parameter `P_VERSION_TYPE` if cost and revenue are planned separately.

- You cannot use this procedure to recalculate amounts for a working version under the following conditions:
  - If the option to allow edit after initial baseline is not enabled for the financial plan type, then you cannot modify a working version after you create a baseline.
  - A budget or forecast version is locked by another user or is locked for processing
  - A budget or forecast version has processing errors
  - A change document version
  - A version for an organization forecast
  - A workplan version

- For forecast versions that are time-phased by periods, this procedure recalculates amounts for lines that do not have actual amounts (where the value of the parameter `P_BUDGET_START_DATE` is after the ETC start date). If a forecast version is not time-phased by periods, then this procedure recalculates amounts for all lines, including lines with actual amounts.

- If the Budget Calculation Extension procedures cannot calculate a value, or calculates a zero amount, then the procedure reports an error.

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- `P_PM_PRODUCT_CODE`
**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, or financial plan type.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes, page 2-22.

**Business Rules**

- The following parameters are used for planning with financial plan types. These parameters are not used for planning with budget types.
  - Plan type identifier and plan type name 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 create a baseline. 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 create a baseline.

- 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. PRESOURCE_LIST_NAME links to PRESOURCE_LIST_ID.
  - For budget lines P_PM_TASK_REFERENCE links to P_PA_TASK_ID. PRESOURCE_ALIAS links to PRESOURCE_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 or financial planning option:

- 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 or financial planning option.

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 or financial planning option.

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 you budget by date range, you must provide the budget start and end dates. These dates may not overlap for a certain resource assignment.

**Note:** You cannot budget by date range for budgets that you create for financial plan types.

The task level at which you pass budget information should correspond to the level specified in the budget entry method. For example, for budgets that you create for budget types, 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.

<table>
<thead>
<tr>
<th>BEM Flag</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost_quantity_flag</td>
<td>quantity</td>
</tr>
<tr>
<td>raw_cost_flag</td>
<td>raw_cost</td>
</tr>
<tr>
<td>burdened_cost_flag</td>
<td>burdened_cost</td>
</tr>
<tr>
<td>rev_quantity_flag</td>
<td>quantity</td>
</tr>
</tbody>
</table>
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 or financial plan type.

You can specify a value for the parameter P_REVENUE_AMOUNT only for a revenue budget, as defined by the budget type or financial plan 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.

If a working budget or forecast version exists for a project and financial plan type, creating a new working budget or forecast version with P_CREATE_NEW_CURR_WORKING_FLAG set to N creates a new working budget or forecast version with budget lines.

If a current working budget or forecast version exists for a project and financial plan type, creating a new working budget or forecast version with P_CREATE_NEW_CURR_WORKING_FLAG set to Y marks the newly created budget or forecast version as Current Working.

If a current working budget or forecast version exists for a project and financial plan type, creating a new working budget or forecast version with P_REPLACE_CURRENT_WORKING_FLAG set to Y deletes the existing Current Working budget or forecast version and budget lines, and marks the newly created budget or forecast version as Current Working.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_BUDGET_TYPE_CODE
CREATE_DRAFT_FINPLAN

CREATE_DRAFT_FINPLAN creates draft budgets and forecasts for financial plan types and is similar to CREATE_DRAFT_BUDGET. This procedure accepts summary data at the project, task, resource, and currency levels. For budget and forecast versions that are time-phased by PA or GL period, the API also spreads the data, including quantities and amounts, across periods based on the spread curve associated with a resource. No other edit or modification of data is performed by this procedure. The validations and function security for this procedure are similar to that for CREATE_DRAFT_BUDGET.

Note: This procedure calls the Budget Calculation Client Extension. If you have enabled the Budget Calculation Client Extension, the client extension may override the following amounts generated by this API procedure:

- raw cost
- burdened cost
- revenue

For additional information on the Budget Calculation Client Extension, see: Budget Calculation Extensions, page 12-3.

Business Rules

- A draft budget requires approval before you can create a baseline. 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 create a baseline for the budget.

- We establish the following links between information stored in your system and certain information in Oracle Projects. Therefore, 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 resource:

- PRESOURCE_LIST_NAME and PRESOURCE_LIST_ID
- RESOURCE_ALIAS and RESOURCE_LIST_MEMBER_ID

If the financial plan type associated with the project specifies that cost and revenue are planned separately, then you must specify whether the budget or forecast version is to be created as a cost version or a revenue version. Otherwise, you do not need to pass the value for P_VERSION_TYPE. If passed, the value should correspond to the value defined for the financial plan type.

You can specify whether planning in multiple currencies is enabled for the budget or forecast version using P_PLAN_IN_MULTI_CURR_FLAG. If this parameter is not passed, then the system uses the value defined for the financial plan type.

You can specify whether the budget or forecast version is time-phased by GL or PA periods, or is non-time phased. If this parameter is not passed, then the system uses the value defined for the financial plan type.

You can specify the planning level of the budget or forecast. If this parameter is not passed, then the system uses the value defined for the financial plan type.

If the plan amounts created using this API are not classified using a resource list, then you must set the value of P_USING_RESOURCE_LISTS_FLAG to N.

When the amount entry fields shown in the following table are set to N, do not pass the related parameters. In this case, the budget or forecast is created with the corresponding amount entry options and edit of these quantities and amounts is not possible.

<table>
<thead>
<tr>
<th>Amount Entry Field</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost_quantity_flag</td>
<td>quantity (used when the financial plan version is for cost only)</td>
</tr>
<tr>
<td>rev_quantity_flag</td>
<td>quantity (used when the financial plan version is for revenue only)</td>
</tr>
<tr>
<td>all_quantity_flag</td>
<td>quantity (used when cost and revenue are planned together in the same financial plan version)</td>
</tr>
<tr>
<td>raw_cost_flag</td>
<td>raw_cost</td>
</tr>
<tr>
<td>Amount Entry Field</td>
<td>Related Parameter</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>burdened_cost_flag</td>
<td>burdened_cost</td>
</tr>
<tr>
<td>revenue_flag</td>
<td>revenue</td>
</tr>
</tbody>
</table>

- You can specify values for the parameters `P_RAW_COST` and `P_BURDENED_COST` only for budgets or forecasts for which the financial plan type allows planning for cost, or planning for cost and revenue together.

- You can specify a value for the parameter `P_REVENUE_AMOUNT` only for a budget or forecast for which the financial plan type allows planning for revenue, or planning for cost and revenue together.

- If descriptive Flexfields are defined for a budget version, you can pass them in as parameters.

- Passing the PL/SQL table `P_FINPLAN_TRANS_TAB` is optional.

- If a budget or forecast is time-phased by PA or GL periods, then the budget or forecast amounts passed to this API are spread using the spread curve associated with a resource between the start and end dates specified in the `P_FINPLAN_TRANS_TAB` composite PL/SQL table parameter.

- You can specify a value for the `PA_TASK_ID` or `PM_TASK_REFERENCE` parameter only when budgeting by tasks, as defined by the planning level of the budget or forecast.

- The task level at which you pass budget information should correspond to the level specified in the planning level of a budget or forecast. For example, if the planning level specifies that you can enter a budget at the top task level, then this API passes top tasks and project level budget and forecast amounts.

- If descriptive Flexfields are defined for a resource assignment, you can pass them in as parameters.

- A draft budget does not require you to create budget lines simultaneously.

- For budgets or forecasts created using financial plan types, multiple working versions can exist along with one current working version. The `P_CREATE_NEW_CURR_WORKING_FLAG` and `P_REPLACE_CURRENT_WORKING_FLAG` parameters determine whether the new budget version is created as a current working version and whether the new version replaces (deletes) the existing current working version.
• If a working budget or forecast version exists for a project and financial plan type, creating a new working budget or forecast version with P_CREATE_NEW_CURR_WORKING_FLAG set to N creates a new working budget or forecast version with budget lines.

• If a current working budget or forecast version exists for a project and financial plan type, creating a new working budget or forecast version with P_CREATE_NEW_CURR_WORKING_FLAG set to Y marks the newly created budget or forecast version as Current Working.

• If a current working budget or forecast version exists for a project and financial plan type, creating a new working budget or forecast version with P_REPLACE_CURRENT_WORKING_FLAG set to Y deletes the existing Current Working budget or forecast version and budget lines, and marks the newly created budget or forecast version as Current Working.

• All business rules for creating a budget or forecast version for a financial plan type are applicable when using this API. For example, if the Current Working version is locked by another user, and P_CREATE_NEW_CURR_WORKING_FLAG or P_REPLACE_CURRENT_WORKING_FLAG parameters are set to Y, then an appropriate message is returned that the Current Working version is locked and cannot be replaced or deleted.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

• P_PM_PRODUCT_CODE

• P_BUDGET_VERSION_NAME

DELETE_BASELINE_BUDGET

DELETE_BASELINE_BUDGET is a PL/SQL procedure used to delete a baseline version in Oracle Projects for either a project and budget type, or a project and financial plan type.

Business Rules

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

General Rules

• For budgets that do not use budgetary controls, you can use this procedure to delete any baseline version except the following:
• Current (latest version)

• Current Original

• For budgets that use budgetary controls, you can use this procedure to delete any baseline version except the following:
  • Current (latest version)
  • If budgetary control balances exist, the version previous to the Current version
  • Current Original

• You cannot use this procedure to delete draft or working versions and submitted versions.

• Oracle Projects establishes the following links between information stored in your system and certain information in Oracle Projects. You can use the following parameters instead of their corresponding Oracle Projects identification codes:
  • P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
  • P_FIN_PLAN_TYPE_NAME links to P_FIN_PLAN_TYPE_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.

• To delete a baseline version and its corresponding lines, you must specify a value for the parameter P_VERSION_NUMBER.
  • If you specify a value for a current original or current baseline version, then the procedure will abort.
  • If budgetary controls are enabled and you specify a value for a previous baseline version that has budgetary control balances, then the procedure will abort.

• Oracle Projects function security controls deletion of the following classes of baseline versions. These baseline classes apply to versions for budget types and financial plan types:
  • Approved Budget (Cost or Revenue): Financials: Project: Approved Budget: Delete Baseline Version
  • Budget (not approved cost or revenue): Financials: Project: Budget: Delete Baseline Version
- Forecast (applicable to financial planning only): Financials: Project: Forecast: Delete Approved Version

**Budget Types**

- The following parameters are not used for planning with budget types:
  - P_FIN_PLAN_TYPE_ID
  - P_FIN_PLAN_TYPE_NAME
  - P_VERSION_TYPE

- You cannot use this procedure to delete a baseline version for a forecast budget type.

**Financial Plan Types**

- All business rules for deleting a baseline version for a financial plan type in the user interface are applicable when you use this procedure.

- The following parameter is not used for planning with financial plan types:
  - P_BUDGET_TYPE_CODE

- You must specify a value for the parameter P_VERSION_TYPE if cost and revenue are planned separately.

- You cannot use this procedure to delete a baseline version under the following conditions:
  - A version for an organization forecast
  - A workplan version

**Parameters**

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

The parameters that identify the version are listed below:

- P_PA_PROJECT_ID
• P_BUDGET_TYPE_CODE
• P_FIN_PLAN_TYPE_ID
• P_VERSION_TYPE
• P_VERSION_NUMBER

DELETE_BUDGET_LINE

DELETE_BUDGET_LINE is a PL/SQL procedure used to delete a line from a draft or working version in Oracle Projects for either a project and budget type, or a project and financial plan type.

Business Rules

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

General Rules

• After you use DELETE_BUDGET_LINE to delete one or more lines from a draft or working version, save the data before you call the procedure BASELINE_BUDGET. (A working version may require approval before you can create a baseline.)

• You can delete lines only from draft or working versions. You cannot delete lines from a submitted or baseline version. You cannot delete lines from an approved revenue budget working version if the autobaseline feature is enabled.

• Oracle Projects establishes the following links between information stored in your system and certain information in Oracle Projects. You can use the following parameters instead of their corresponding Oracle Projects identification codes:
  • P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
  • P_FIN_PLAN_TYPE_NAME links to P_FIN_PLAN_TYPE_ID.
  • P_PM_TASK_REFERENCE links to P_PA_TASK_ID.
  • PRESOURCE_ALIAS links to PRESOURCE_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.

• You do not need to specify values for the following pair of parameters if the version is not categorized by resources:
• PRESOURCE_ALIAS and PRESOURCE_LIST_MEMBER_ID

• Oracle Projects does not validate a date value specified for the parameter P_START_DATE.

• If a version is time-phased by PA periods, then the value specified for the parameter P_PERIOD_NAME must map to a PA period. Likewise, if a version is time-phased by GL periods, then the value specified for the parameter P_PERIOD_NAME must map to a GL period. If the value specified for P_PERIOD_NAME is invalid, then the procedure will abort.

• All business rules for deleting a line from a version for a budget type or a financial plan type in the user interface are applicable when you use this procedure.

**Budget Types**

• The following parameters are not used for planning with budget types:
  • P_FIN_PLAN_TYPE_ID
  • P_FIN_PLAN_TYPE_NAME
  • P_VERSION_NUMBER
  • P_VERSION_TYPE
  • P_CURRENCY_CODE

• You cannot use this procedure to delete a budget line for a forecast budget type.

• If no values are specified for the parameters P_START_DATE and P_PERIOD_NAME, then the procedure deletes all budget lines for a task/resource combination.

• If no value is specified for the parameter P_PERIOD_NAME, then the procedure uses the parameter P_START_DATE to determine the period.

• If values are specified for the parameter P_START_DATE and the parameter P_PERIOD_NAME, then the procedure uses P_PERIOD_NAME to derive the start date.

• The following rules apply to the budget entry method:
  • Depending on the budget entry method, this procedure may require you to specify task and resource data.

• If no task data is specified, then Oracle Projects assumes that the budget entry
method specifies uncategorized budgeting (budgets not tracked by resource) and project-level budgeting.

- If the budget entry method for the version is not time-phased by PA or GL periods, and values are specified for either the parameter P_START_DATE or the parameter P_PERIOD_NAME, then the procedure will abort.

- For date-range budgets, the parameter P_START_DATE corresponds to the start date of the budget line date range. For budgets time-phased by PA or GL periods, the parameter, P_START_DATE corresponds to the start date of the period for which the budget line is defined.

**Financial Plan Types**

- The following parameter is not used for planning with financial plan types:
  - P_BUDGET_TYPE_CODE

- If no value is specified for the parameter P_VERSION_NUMBER, then this procedure deletes a line from the current working version. You must provide a value for this parameter to delete a line from a version other than the current working version.

- You must specify a value for the parameter P_VERSION_TYPE if cost and revenue are planned separately.

- You cannot use this procedure to delete a line under the following conditions:
  - If the option to allow edit after initial baseline is not enabled for the financial plan type, then you cannot modify a working version after you create a baseline.
  - A budget or forecast version is locked by another user or is locked for processing
  - A budget or forecast version has processing errors
  - A change document version
  - A version for an organization forecast
  - A workplan version

- Depending on the planning options defined for the version, this procedure may require you to specify task and resource data.

- You must specify a value for the parameter P_CURRENCY_CODE if you are
planning in multiple transaction currencies.

- If no values are specified for the parameters P_CURRENCY_CODE, P_START_DATE, and P_PERIOD_NAME, then the procedure deletes all lines for the specified task/resource combination.

- If you specify a value for the parameter P_CURRENCY_CODE, and do not specify values for the parameters P_START_DATE and P_PERIOD_NAME, then the procedure deletes all lines for the specified currency code and task/resource combination.

- If no value is specified for the parameter P_PERIOD_NAME, then the procedure uses the value specified for the parameter P_START_DATE to determine the period.

- If values are specified for the parameters P_START_DATE and P_PERIOD_NAME, then the procedure uses P_PERIOD_NAME to derive the start date.

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

The parameters that identify the version and line are listed below:

- P_PA_PROJECT_ID
- P_BUDGET_TYPE_CODE
- P_FIN_PLAN_TYPE_ID
- P_VERSION_TYPE
- P_VERSION_NUMBER
- P_PA_TASK_ID
- P_RESOURCE_LIST_MEMBER_ID
- P_CURRENCY_CODE
- P_START_DATE
- P_PERIOD_NAME
DELETE_DRAFT_BUDGET

DELETE_DRAFT_BUDGET is a PL/SQL procedure used to delete a draft or working version in Oracle Projects for either a project and budget type, or a project and financial plan type.

Business Rules

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

General Rules

• You can delete only draft or working versions. You cannot delete a submitted or baseline version. You cannot delete an approved revenue budget working version if the autobaseline feature is enabled.

• Oracle Projects establishes the following links between information stored in your system and certain information in Oracle Projects. You can use the following parameters instead of their corresponding Oracle Projects identification codes:
  • P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
  • P_FIN_PLAN_TYPE_NAME links to P_FIN_PLAN_TYPE_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.

• All business rules for deleting a draft or working version for a budget type or a financial plan type in the user interface are applicable when you use this procedure.

Budget Types

• The following parameters are not used for planning with budget types:
  • P_FIN_PLAN_TYPE_ID
  • P_FIN_PLAN_TYPE_NAME
  • P_VERSION_NUMBER
  • P_VERSION_TYPE

• You cannot use this procedure to delete a working version for a forecast budget type.
Financial Plan Types

- The following parameter is not used for planning with financial plan types:
  - P_BUDGET_TYPE_CODE

- If no value is specified for the parameter P_VERSION_NUMBER, then this procedure deletes the current working version. You must provide a value for this parameter to delete a version other than the current working version.

- You must specify a value for the parameter P_VERSION_TYPE if cost and revenue are planned separately.

- You cannot use this procedure to delete a working version under the following conditions:
  - If the option to allow edit after initial baseline is not enabled for the financial plan type, then you cannot delete a working version after you create a baseline.
  - A budget or forecast version is locked by another user or is locked for processing
  - A budget or forecast version has processing errors
  - A change document version
  - A version for an organization forecast
  - A workplan version

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

Parameter that identify the version are listed below:

- P_PA_PROJECT_ID
- P_BUDGET_TYPE_CODE
- P_FIN_PLAN_TYPE_ID
- P_VERSION_TYPE
• P_VERSION_NUMBER

GET_PROJECT_ID

To retrieve information about all the financial plan types attached to a project, the project context must be set. GET_PROJECT_ID returns the PROJECT_ID for the project in context used by the public view PA_FINPLAN_TYPES_V. This view is used by Oracle Project Connect.

SET_PROJECT_ID

SET_PROJECT_ID used to set the public variable G_PROJECT_ID which is used by public view PA_FINPLAN_TYPES_V. This view is used by Oracle Project Connect.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The required parameters for SET_PROJECT_ID are listed below:
• P_PROJECT_ID

UPDATE_BUDGET

UPDATE_BUDGET is a PL/SQL procedure used to update a draft or working version in Oracle Projects for either a project and budget type, or a project and financial plan type. This procedure updates existing lines or inserts new lines, depending on whether lines already exist.

This API uses composite datatypes. For more information, see APIs That Use Composite Datatypes, page 2-22.

Business Rules

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

General Rules

• After you use UPDATE_BUDGET to update a draft or working version, save the data before you call the procedure BASELINE_BUDGET. (A working version may require approval before you can create a baseline.)

• You can update only draft or working versions. You cannot update submitted or baseline versions. You cannot update an approved revenue budget working version if the autobaseline feature is enabled.

• With respect to lines, you can use this procedure only to update existing lines or to
add lines. To delete existing lines, use the procedure DELETE_BUDGET_LINE.

- Oracle Projects establishes the following links between information stored in your system and certain information in Oracle Projects. You can use the following parameters instead of their corresponding Oracle Projects identification codes:
  - P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
  - P_FIN_PLAN_TYPE_NAME links to P_FIN_PLAN_TYPE_ID.
  - PM_TASK_REFERENCE links to PA_TASK_ID.
  - RESOURCE_ALIAS links to 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.

- You do not need to specify values for the following pair of parameters if the version is not categorized by resources:
  - RESOURCE_ALIAS and RESOURCE_LIST_MEMBER_ID

- If a version is time-phased by period, then Oracle Projects identifies a line either by the start date or period name. Therefore, you cannot change start dates and period names for lines.

- The following rules apply for versions that are time-phased by PA or GL periods:
  - If a valid value is specified for the parameter PERIOD_NAME, then this procedure updates the line for the specified period.
    If a version is time-phased by PA periods, then the value specified for the parameter PERIOD_NAME must map to a PA period. Likewise, if a version is time-phased by GL periods, then the value specified for PERIOD_NAME must map to a GL period.
    If the value specified for PERIOD_NAME is invalid, then the procedure will abort.
  - If values are specified for the parameter BUDGET_START_DATE and the parameter PERIOD_NAME, then the procedure uses PERIOD_NAME to determine the period.
  - If no value is specified for the parameter PERIOD_NAME, then the procedure uses the parameter BUDGET_START_DATE to determine the period.
    If a version is time-phased by PA periods, then the value for the parameter BUDGET_START_DATE must map to the start date of a PA period. Likewise, if
a version is time-phased GL periods, then the value for the parameter BUDGET_START_DATE must map to the start date of a GL period.

If Oracle Projects cannot determine a valid period name, then the procedure will abort.

- If no values are specified for the parameter BUDGET_START_DATE and the parameter PERIOD_NAME, then the procedure will abort.

- If Descriptive Flexfields are defined, then you can pass them as IN parameters.

- All business rules for updating a version for a budget type or a financial plan type in the user interface are applicable when you use this procedure.

**Budget Types**

- The following parameters are not used for planning with budget types:
  - P_FIN_PLAN_TYPE_ID
  - P_FIN_PLAN_TYPE_NAME
  - P_BUDGET_VERSION_NUMBER
  - P_VERSION_TYPE
  - TXN_CURRENCY_CODE
  - All currency conversion attributes

- You cannot use this procedure to update a budget version or budget lines for a forecast budget type.

- Specify values for the parameters PA_TASK_ID or PM_TASK_REFERENCE only when planning by tasks, as defined by the budget entry method.

- Specify values for the parameter PERIOD_NAME only when planning by PA or GL period, as defined by the budget entry method.

- The task level at which you specify 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 procedure should pass only lowest tasks.

- When the budget entry method (BEM) restricts entry of a certain field as shown in the following table, do not specify values for the corresponding parameter.
<table>
<thead>
<tr>
<th>BEM Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost quantity not enterable</td>
<td>quantity</td>
</tr>
<tr>
<td>Raw cost not enterable</td>
<td>raw_cost</td>
</tr>
<tr>
<td>Burdened cost not enterable</td>
<td>burdened_cost</td>
</tr>
<tr>
<td>Revenue quantity not enterable</td>
<td>quantity</td>
</tr>
<tr>
<td>Revenue not enterable</td>
<td>revenue</td>
</tr>
</tbody>
</table>

- If a project and a budget type are enabled for budgetary controls, then you cannot update or add budget lines for GL periods that are after the latest encumbrance year defined in the ledger associated with the project.

**Financial Plan Types**

- The following parameter is not used for planning with financial plan types:

  - P_BUDGET_TYPE_CODE

- If no value is specified for the parameter P_BUDGET_VERSION_NUMBER, then this procedure updates a line for the current working version.

- You can specify values for the parameters PA_TASK_ID or PM_TASK_REFERENCE only when planning by tasks, as defined by the planning level of the budget or forecast.

- The task level at which you specify plan information should correspond to the level specified in the planning level of a budget or forecast version. For example, if the planning level specifies that you can enter a budget or forecast at the top task level, then this procedure should pass only top task and project-level budget and forecast amounts.

- You cannot use this procedure to update a working version under the following conditions:

  - If the option to allow edit after initial baseline is not enabled for the financial plan type, then you cannot modify a working version after you create a baseline.

  - A budget or forecast version is locked by another user or is locked for processing.
• A budget or forecast version has processing errors
• A change document version
• A version for an organization forecast
• A workplan version

• When the amount entry options restrict entry of certain fields as shown in the following table, do not specify values for the corresponding parameters. In these cases, the budget or forecast is created with the specified amount entry options and edit of the corresponding quantities and amounts is not allowed.

<table>
<thead>
<tr>
<th>Amount Entry Option Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost quantity not enterable</td>
<td>quantity (used when the version is for cost only)</td>
</tr>
<tr>
<td>Revenue quantity not enterable</td>
<td>quantity (used when the version is for revenue only)</td>
</tr>
<tr>
<td>Quantity not enterable</td>
<td>quantity (used when cost and revenue are planned together in the same version)</td>
</tr>
<tr>
<td>Raw cost not enterable</td>
<td>raw_cost</td>
</tr>
<tr>
<td>Burdened cost not enterable</td>
<td>burdened_cost</td>
</tr>
<tr>
<td>Revenue not enterable</td>
<td>revenue</td>
</tr>
</tbody>
</table>

• You can specify values for the parameters RAW_COST and BURDENED_COST only for budgets or forecasts for which the financial plan type allows planning for cost, or for cost and revenue together in the same version.

• You can specify a value for the parameter REVENUE only for a budget or forecast for which the financial plan type allows planning for revenue, or for cost and revenue together in the same version.

Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

• P_API_VERSION_NUMBER
• P_PM_PRODUCT_CODE

Parameter in this procedure that identify the version and line are listed below:
• P_PA_PROJECT_ID
• P_BUDGET_TYPE_CODE
• P_FINPLAN_TYPE_ID
• P_VERSION_TYPE
• P_BUDGET_VERSION_NUMBER

Table of Record: P_BUDGET_LINES_IN

The following table describes the data structure for the parameter P_BUDGET_LINES_IN shown in the parameters table.

Parameter names marked with an asterisk (*) in the following table identify the version and line.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_TASK_ID*</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Identifier for the task in Oracle Projects</td>
</tr>
<tr>
<td>PM_TASK_REFERENCE*</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The reference code that uniquely identifies the task in the external system</td>
</tr>
<tr>
<td>RESOURCE_ALIAS*</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Alias of a resource uniquely identifies the task in the external system</td>
</tr>
<tr>
<td>RESOURCE_LIST_MEMBER_ID*</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>The identification code of the resource</td>
</tr>
<tr>
<td>TXN_CURRENCY_CODE*</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Financial plan currency identifier. Required if planning in multiple transaction currencies.</td>
</tr>
<tr>
<td>PERIOD_NAME*</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>GL or PA period name</td>
</tr>
<tr>
<td>BUDGET_START_DATE*</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>Start date of budget line</td>
</tr>
<tr>
<td>BUDGET_END_DATE*</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>End date of budget line</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------</td>
<td>------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QUANTITY</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted quantity</td>
</tr>
<tr>
<td>RAW_COST</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted raw cost amount</td>
</tr>
<tr>
<td>BURDENED_COST</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted burdened cost amount</td>
</tr>
<tr>
<td>REVENUE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Budgeted revenue amount</td>
</tr>
<tr>
<td>CHANGE_REASON_CODE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Code that identifies the change reason for the line</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Line description</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project functional currency cost rate type for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_DATE_TYP</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project functional currency cost rate date type for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_COST_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>Project functional currency cost rate date for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_COST_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Project functional currency cost exchange rate for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_REV_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project functional currency revenue rate type for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_REV_RATE_DATE_TYP</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project functional currency revenue rate date type for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_REV_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>Project functional currency revenue rate date for financial plan types</td>
</tr>
<tr>
<td>PROJFUNC_REV_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Project functional currency revenue exchange rate for financial plan types</td>
</tr>
<tr>
<td>Name</td>
<td>Usage</td>
<td>Type</td>
<td>Req?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------</td>
<td>------------</td>
<td>------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>PROJECT_COST_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project currency cost rate type for financial plan types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_COST_RATE_DATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project currency cost rate date type for financial plan types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_COST_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>Project currency cost rate date for financial plan types</td>
</tr>
<tr>
<td>PROJECT_COST_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Project currency cost exchange rate for financial plan types</td>
</tr>
<tr>
<td>PROJECT_REV_RATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project currency revenue rate type for financial plan types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_REV_RATE_DATE_TYPE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Project currency revenue rate date type for financial plan types</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT_REV_RATE_DATE</td>
<td>IN</td>
<td>DATE</td>
<td>No</td>
<td>Project currency revenue rate date for financial plan types</td>
</tr>
<tr>
<td>PROJECT_REV_EXCHANGE_RATE</td>
<td>IN</td>
<td>NUMBER</td>
<td>No</td>
<td>Project currency revenue exchange rate for financial plan types</td>
</tr>
<tr>
<td>PM_PRODUCT_CODE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>The product code of the vendor of the external system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM_BUDGET_LINE_REFERENCE</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Reference code that identifies the budget line on the client side</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE_CATEGORY</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Used by Descriptive Flexfields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE1 THROUGH ATTRIBUTE15</td>
<td>IN</td>
<td>VARCHAR2</td>
<td>No</td>
<td>Budget line Descriptive Flexfields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(150)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table of Record: P_BUDGET_LINES_OUT**

The following table shows the data structure for the parameter
**P_BUDGET_LINES_OUT** shown in the parameters table.

<table>
<thead>
<tr>
<th>Name</th>
<th>Usage</th>
<th>Type</th>
<th>Req?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN_STATUS</td>
<td>IN</td>
<td>VARCHAR2(1)</td>
<td>No</td>
<td>Return status</td>
</tr>
</tbody>
</table>

**UPDATE_BUDGET_LINE**

UPDATE_BUDGET_LINE is a PL/SQL procedure used to update a line for a draft or working version in Oracle Projects for either a project and budget type, or a project and financial plan type.

**Business Rules**

This section describes general business rules, and rules for using this procedure with versions created for budget types and for financial plan types.

**General Rules**

- After you use UPDATE_BUDGET_LINE to update a line for a draft or working version, save the data before you call the procedure BASELINE_BUDGET. (A working version may require approval before you can create a baseline.)

- You can update lines only for draft or working versions. You cannot update lines for submitted or baseline versions. You cannot update lines for an approved revenue budget working version if the autobaseline feature is enabled.

- Oracle Projects establishes the following links between information stored in your system and certain information in Oracle Projects. You can use the following parameters instead of their corresponding Oracle Projects identification codes:

  - P_PM_PROJECT_REFERENCE links to P_PA_PROJECT_ID.
  - P_FIN_PLAN_TYPE_NAME links to P_FIN_PLAN_TYPE_ID.
  - 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.

- You do not need to specify values for the following pair of parameters if the version is not categorized by resources:
• P_RESOURCE_ALIAS and P_RESOURCE_LIST_MEMBER_ID

• If a version is time-phased by period, then Oracle Projects identifies a line either by the start date or period name. Therefore, you cannot change start dates and period names for lines.

• The following rules apply for versions that are time-phased by PA or GL periods:
  • If a valid value is specified for the parameter P_PERIOD_NAME, then this procedure updates the line for the specified period.
  If a version is time-phased by PA periods, then the value specified for the parameter P_PERIOD_NAME must map to a PA period. Likewise, if a version is time-phased by GL periods, then the value specified for P_PERIOD_NAME must map to a GL period.
  If the value specified for P_PERIOD_NAME is invalid, then the procedure will abort.

  • If values are specified for the parameter P_BUDGET_START_DATE and the parameter P_PERIOD_NAME, then the procedure uses P_PERIOD_NAME to determine the period.
  If a version is time-phased by PA periods, then the value for the parameter P_BUDGET_START_DATE must map to the start date of a PA period. Likewise, if a version is time-phased GL periods, then the value for the parameter P_BUDGET_START_DATE must map to the start date of a GL period.
  If Oracle Projects cannot determine a valid period name, then the procedure will abort.

  • If no values are specified for the parameter P_BUDGET_START_DATE and the parameter P_PERIOD_NAME, then the procedure will abort.

• If Descriptive Flexfields are defined, then you can pass them as IN parameters.

• All business rules for updating lines for a budget type or a financial plan type in the user interface are applicable when you use this procedure.

**Budget Types**

• The following parameters are not used for planning with budget types:
  • P_FIN_PLAN TYPE_ID
• P_FIN_PLAN_TYPE_NAME
• P_VERSION_NUMBER
• P_VERSION_TYPE
• P_CURRENCY_CODE
• All currency conversion attributes

• You cannot use this procedure to update budget lines for a forecast budget type.

• Specify values for the parameters P_PA_TASK_ID or P_PM_TASK_REFERENCE only when planning by tasks, as defined by the budget entry method.

• Specify values for the parameter P_PERIOD_NAME only when planning by PA or GL period, as defined by the budget entry method.

• The task level at which you specify 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 procedure should pass only lowest tasks.

• When the budget entry method (BEM) restricts entry of a certain field as shown in the following table, do not specify values for the corresponding parameter.

<table>
<thead>
<tr>
<th>BEM Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost quantity not enterable</td>
<td>p_quantity</td>
</tr>
<tr>
<td>Raw cost not enterable</td>
<td>p_raw_cost</td>
</tr>
<tr>
<td>Burdened cost not enterable</td>
<td>p_burdened_cost</td>
</tr>
<tr>
<td>Revenue quantity not enterable</td>
<td>p_quantity</td>
</tr>
<tr>
<td>Revenue not enterable</td>
<td>p_revenue</td>
</tr>
</tbody>
</table>

• If a project and a budget type are enabled for budgetary controls, then you cannot update or add budget lines for GL periods that are after the latest encumbrance year defined in the ledger associated with the project.
Financial Plan Types

- The following parameter is not used for planning with financial plan types:
  - P_BUDGET_TYPE_CODE

- If no value is specified for the parameter P_VERSION_NUMBER, then this procedure updates a line for the current working version.

- You can specify values for the parameters P_PA_TASK_ID or P_PM_TASK_REFERENCE only when planning by tasks, as defined by the planning level of the budget or forecast.

- The task level at which you specify plan information should correspond to the level specified in the planning level of a budget or forecast version. For example, if the planning level specifies that you can enter a budget or forecast at the top task level, then this procedure should pass only top task and project-level budget and forecast amounts.

- You cannot use this procedure to update a working version under the following conditions:
  - If the option to allow edit after initial baseline is not enabled for the financial plan type, then you cannot modify a working version after you create a baseline.
  - A budget or forecast version is locked by another user or is locked for processing
  - A budget or forecast version has processing errors
  - A change document version
  - A version for an organization forecast
  - A workplan version

- When the amount entry options restrict entry of certain fields as shown in the following table, do not specify values for the corresponding parameters. In these cases, the budget or forecast is created with the specified amount entry options and edit of the corresponding quantities and amounts is not allowed.

<table>
<thead>
<tr>
<th>Amount Entry Option Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost quantity not enterable</td>
<td>p_quantity (used when the version is for cost only)</td>
</tr>
</tbody>
</table>
### Amount Entry Option Setting

<table>
<thead>
<tr>
<th>Amount Entry Option Setting</th>
<th>Related Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenue quantity not enterable</td>
<td>p_quantity (used when the version is for revenue only)</td>
</tr>
<tr>
<td>Quantity not enterable</td>
<td>p_quantity (used when cost and revenue are planned together in the same version)</td>
</tr>
<tr>
<td>Raw cost not enterable</td>
<td>p_raw_cost</td>
</tr>
<tr>
<td>Burdened cost not enterable</td>
<td>p_burdened_cost</td>
</tr>
<tr>
<td>Revenue not enterable</td>
<td>p_revenue</td>
</tr>
</tbody>
</table>

- You can specify values for the parameters P_RAW_COST and P_BURDENED_COST only for budgets or forecasts for which the financial plan type allows planning for cost, or for cost and revenue together in the same version.
- You can specify a value for the parameter P_REVENUE only for a budget or forecast for which the financial plan type allows planning for revenue, or for cost and revenue together in the same version.

### Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters in this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE

The parameters in this procedure that identify the version and line are listed below:

- P_PA_PROJECT_ID
- P_BUDGET_TYPE_CODE
- P_FIN_PLAN_TYPE_ID
- P_VERSION_TYPE
- P_VERSION_NUMBER
- P_PA_TASK_ID
• P_RESOURCE_LIST_MEMBER_ID
• P_CURRENCY_CODE
• P_BUDGET_START_DATE
• P_PERIOD_NAME

CLEAR_BUDGET

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

CLEAR_CALCULATE_AMOUNTS

CLEAR_CALCULATE_AMOUNTS is a Load-Execute-Fetch procedure used to reset the global data structures used by the Load-Execute-Fetch procedure CALCULATE_AMOUNTS. The procedure CALCULATE_AMOUNTS calls CLEAR_CALCULATE_AMOUNTS to initialize the data structure that is used to fetch the calculated amounts for a plan version.

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, or financial plan type. For each budget line, this API writes to globals that can be read by the API FETCH_CALCULATE_AMOUNTS.

Business Rules

• The following parameters are used only for budgets that are created for a financial plan type:
  • P_BUDGET_VERSION_ID
  • P_FIN_PLAN_TYPE_ID
  • P_FIN_PLAN_TYPE_NAME
  • P_VERSION_TYPE
  • P_BUDGET_VERSION_NUMBER

• Because 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for EXECUTE_CALCULATE_AMOUNTS are listed below:

• P_API_VERSION_NUMBER

• P_PM_PRODUCT_CODE

• P_BUDGET_TYPE_CODE

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

The following parameters are used for planning with financial plan types. These parameters are not used for planning with budget types.

• Plan type identifier and plan type name 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
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for EXECUTE_CREATE_DRAFT_BUDGET are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_BUDGET_TYPE_CODE
- P_ENTRY_METHOD_CODE

EXECUTE_CREATE_DRAFT_FINPLAN

EXECUTE_CREATE_DRAFT_FINPLAN creates budgets and forecasts using the data stored in the global tables during the Load process. Before calling this procedure, you should call LOAD_RESOURCE_INFO to load the resource information along with the quantity and amounts required for budget and forecast line creation.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for EXECUTE_CREATE_DRAFT_FINPLAN are listed below:

- P_PM_PRODUCT_CODE
- P_BUDGET_VERSION_NAME

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for EXECUTE_UPDATE_BUDGET are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_BUDGET_TYPE_CODE
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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for FETCH_BUDGET_LINE are listed below:

- P_API_VERSION_NUMBER
- P_LINE_INDEX

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.

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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for FETCH_CALCULATE_AMOUNTS are listed below:

- P_API_VERSION_NUMBER

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.

The following parameters are used for planning with financial plan types. These
parameters are not used for planning with budget types.

- P_TXN_CURRENCY_CODE
- Currency attributes
- P_CHANGE_REASON_CODE

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for LOAD_BUDGET_LINE are listed below:

- P_API_VERSION_NUMBER
- P_TXN_CURRENCY_CODE

LOAD_RESOURCE_INFO

Call LOAD_RESOURCE_INFO before you call EXECUTE_CREATE_DRAFT_FINPLAN to load resource information required to create a budget or forecast. This procedure loads the resource information along with summary amounts to a global PL/SQL table used by EXECUTE_CREATE_DRAFT_FINPLAN.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. There are no required parameters for LOAD_RESOURCE_INFO.

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 6-7.
- Most of the Oracle Projects APIs use a standard set of input and output parameters. See Standard API Parameters, page 2-21.
- 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 6-63.
- 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 6-66.
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 2-8.

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 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:

<table>
<thead>
<tr>
<th>Budget Type Code</th>
<th>Budget Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Approved Cost Budget</td>
</tr>
<tr>
<td>AR</td>
<td>Approved Revenue Budget</td>
</tr>
<tr>
<td>FC</td>
<td>Forecast Cost Budget</td>
</tr>
<tr>
<td>FR</td>
<td>Forecast Revenue Budget</td>
</tr>
</tbody>
</table>

The following 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 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:
<table>
<thead>
<tr>
<th>Amount</th>
<th>LOAD_BUDGET_LINE Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Cost</td>
<td>P_RAW_COST</td>
</tr>
<tr>
<td>Burdened Cost</td>
<td>P_BURDENED_COST</td>
</tr>
<tr>
<td>Cost Quantity</td>
<td>P_QUANTITY</td>
</tr>
<tr>
<td>Revenue</td>
<td>P_REVENUE</td>
</tr>
<tr>
<td>Revenue Quantity</td>
<td>P_QUANTITY</td>
</tr>
</tbody>
</table>

**Select the Budget Entry Method**

Oracle Projects predefines the budget entry methods shown in the following table:

<table>
<thead>
<tr>
<th>Budget Entry Method Code</th>
<th>Budget Entry Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_LOWEST_TASK_BY_PA_PERIOD</td>
<td>By lowest tasks and PA period, categorized by resource</td>
</tr>
<tr>
<td>PA_LOWEST_TASK_BY_GL_PERIOD</td>
<td>By lowest tasks and GL period, categorized by resource</td>
</tr>
<tr>
<td>PA_LOWEST_TASK_BY_DATE_RANGE</td>
<td>By lowest tasks and date range, categorized by resource</td>
</tr>
</tbody>
</table>

The following SQL statement retrieves the budget entry method:

```sql
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
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 SQL statement retrieves the resource list information:

```sql
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:

<table>
<thead>
<tr>
<th>Selected Value</th>
<th>Budget Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOURCE_LIST_ID</td>
<td>P_RESOURCE_LIST_ID</td>
</tr>
<tr>
<td>RESOURCE_LIST_NAME</td>
<td>P_RESOURCE_LIST_NAME</td>
</tr>
</tbody>
</table>

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 SQL statement retrieves the reason for the budget change:

```sql
SELECT code, name
FROM pa_budget_change_reason_v
```

The following table illustrates the relationship between certain selected values and budget parameters:

<table>
<thead>
<tr>
<th>Selected Value</th>
<th>Budget Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODE</td>
<td>PCHANGE_REASON_CODE</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>P_DESCRIPTION</td>
</tr>
</tbody>
</table>

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 SQL statements to retrieve valid task values:

- Budget at the top task level
  ```sql
  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
  ```sql
  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
  ```sql
  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

<table>
<thead>
<tr>
<th>TASK_ID</th>
<th>P_PA_TASK_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM_TASK_REFERENCE</td>
<td>P_PM_TASK_REFERENCE</td>
</tr>
</tbody>
</table>

**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 SQL statement to retrieve the resource list member information:

```sql
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:

<table>
<thead>
<tr>
<th>Selected Value</th>
<th>Budget Line Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOURCE_LIST_MEMBER_ID</td>
<td>PRESOURCE_LIST_MEMBER_ID</td>
</tr>
<tr>
<td>ALIAS</td>
<td>PRESOURCE_ALIAS</td>
</tr>
</tbody>
</table>

**Select Periods**

How the budget entry method is time-phased affects which budget line parameters accept passed values, as shown in the following table:

<table>
<thead>
<tr>
<th>Time-Phased By</th>
<th>Parameters That Accept Values (START_DATE, END_DATE, and PERIOD_NAME)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Time-Phasing</td>
<td>None</td>
</tr>
<tr>
<td>Date Ranges</td>
<td>START_DATE and END_DATE</td>
</tr>
</tbody>
</table>
Time-Phased By Parameters That Accept Values (START_DATE, END_DATE, and PERIOD_NAME)

<table>
<thead>
<tr>
<th>PA or GL Period</th>
<th>START_DATE and END_DATE or PERIOD_NAME</th>
</tr>
</thead>
</table>

When you use time-phased budgeting, you can use the following SQL statements to retrieve the appropriate date information:

- Period name
  
  ```sql
  SELECT period_name
  FROM pa_budget_periods_v
  WHERE period_type_code = &time_phased_type_code
  ```

- Begin and end dates
  
  ```sql
  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:

<table>
<thead>
<tr>
<th>Selected Value</th>
<th>Budget Line Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD_NAME</td>
<td>P_PERIOD_NAME</td>
</tr>
<tr>
<td>PERIOD_START_DATE</td>
<td>P_BUDGET_START_DATE</td>
</tr>
<tr>
<td>PERIOD_END_DATE</td>
<td>P_BUDGET_END_DATE</td>
</tr>
</tbody>
</table>

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.

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 2-25.

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 2-21. 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 6-66.
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_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_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_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;
elif i = 2 THEN
  l_budget_lines_in_rec.resource_list_member_id := 1402;
  l_budget_lines_in_rec.pa_task_id := 2443;
elif i = 3 THEN
  l_budget_lines_in_rec.resource_list_member_id := 1404;
  l_budget_lines_in_rec.pa_task_id := 2446;
elif i = 4 THEN
  l_budget_lines_in_rec.resource_list_member_id := 1407;
  l_budget_lines_in_rec.pa_task_id := 2449;
elif 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 =>
1_api_version_number,
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 =>
1_api_version_number,
p_return_status => l_return_status)
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 6-63.
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);
    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;

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_msg_count => l_msg_count,
        p_msg_index_out => l_msg_index_out );
    dbms_output.put_line ('error mesg ' || l_data);
  end loop;

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_msg_count => l_msg_count,
        p_msg_index_out => l_msg_index_out );
  end loop;
refresh_rates

The refresh rates procedure enables you to refresh conversion rates, cost rates, and bill rates in both workplans and financial plans. The internal name of the procedure is pa_fp_calc_plan_pub.refresh_rates. The procedure is contained in the public API package PA_FP_CALC_PLAN_PUB.

Business Rules

- You must supply a value for either P_PROJECT_ID or P_PM_PROJECT_REFERENCE. (These parameters have "Conditional" indicated in the "Required?" column.)

- You must supply a value for the parameter P_STRUCTURE_VERSION_ID if the
value of P_UPDATE_PLAN_TYPE is WORKPLAN.

- If the value of P_UPDATE_PLAN_TYPE is FINPLAN, then either P_BUDGET_VERSION_ID must have a value, or the following four parameters must all have a value.
  - P_VERSION_TYPE
  - P_BUDGET_VERSION_NUMBER
  - P_FINPLAN_TYPE_ID
  - P_FINPLAN_TYPE_NAME

If P_BUDGET_VERSION_ID is populated, then the other four parameters are ignored.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

- P_API_VERSION_NUMBER
- P_PM_PRODUCT_CODE
- P_PROJECT_ID (conditional -- see Business Rules)
- P_PM_PROJECT_REFERENCE (conditional -- see Business Rules)
- P_UPDATE_PLAN_TYPE
- P_STRUCTURE_VERSION_ID (conditional -- see Business Rules)
- P_BUDGET_VERSION_ID (conditional -- see Business Rules)
- P_BUDGET_VERSION_NUMBER (conditional -- see Business Rules)
- P_VERSION_TYPE (conditional -- see Business Rules)
- P_FINPLAN_TYPE_ID (conditional -- see Business Rules)
- P_FINPLAN_TYPE_NAME (conditional -- see Business Rules)

**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.
Note: Project status inquiry does not report budgets created using financial plan types. Consequently, the status APIs and views related to budgeting and forecasting listed below only report and process budgets and forecasts created using budget types.

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 PARESOURCE_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.
<table>
<thead>
<tr>
<th>Predefined Resource Type</th>
<th>Track as Labor</th>
<th>Unit of Measure</th>
<th>Rollup Actual Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee</td>
<td>Yes</td>
<td>Hours</td>
<td>No</td>
</tr>
<tr>
<td>Job</td>
<td>Yes</td>
<td>Hours</td>
<td>No</td>
</tr>
<tr>
<td>Organization</td>
<td>Yes</td>
<td>Hours</td>
<td>No</td>
</tr>
<tr>
<td>Expenditure Type</td>
<td>Depends on the expenditure type attribute with track as labor</td>
<td>Unit of measure specified for the expenditure type</td>
<td>Yes</td>
</tr>
<tr>
<td>Event Type</td>
<td>No</td>
<td>blank</td>
<td>No</td>
</tr>
<tr>
<td>Supplier</td>
<td>No</td>
<td>blank</td>
<td>No</td>
</tr>
<tr>
<td>Expenditure Category</td>
<td>Depends whether the expenditure category includes a labor expenditure type</td>
<td>Depends whether the resource type is tracked as labor</td>
<td>No</td>
</tr>
<tr>
<td>Revenue Category</td>
<td>Depends whether the revenue category includes a labor expenditure type</td>
<td>Depends whether the resource type is tracked as labor</td>
<td>No</td>
</tr>
</tbody>
</table>

The following table shows the logic of the view as it relates to predefined resources.

<table>
<thead>
<tr>
<th>Predefined Resource</th>
<th>Track as Labor</th>
<th>Unit of Measure</th>
<th>Rollup Actual Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncategorized</td>
<td>Yes</td>
<td>Hours</td>
<td>No</td>
</tr>
<tr>
<td>Unclassified</td>
<td>No</td>
<td>blank</td>
<td>No</td>
</tr>
</tbody>
</table>

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:

```sql
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:

<table>
<thead>
<tr>
<th>List</th>
<th>Units</th>
<th>Hours</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labor</td>
<td>406</td>
<td>406</td>
<td>0</td>
</tr>
<tr>
<td>Senior.Consultant</td>
<td>40</td>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>Principal.Consultant</td>
<td>122</td>
<td>122</td>
<td>0</td>
</tr>
<tr>
<td>Senior.Engineer</td>
<td>164</td>
<td>164</td>
<td>0</td>
</tr>
<tr>
<td>Principal.Engineer</td>
<td>80</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>Travel</td>
<td>4372</td>
<td>0</td>
<td>4372</td>
</tr>
<tr>
<td>Air Travel</td>
<td>3762</td>
<td>0</td>
<td>3762</td>
</tr>
<tr>
<td>Personal Auto Use</td>
<td>105</td>
<td>0</td>
<td>105</td>
</tr>
<tr>
<td>In-House Recoverables</td>
<td>222</td>
<td>0</td>
<td>222</td>
</tr>
<tr>
<td>Computer Services</td>
<td>62</td>
<td>0</td>
<td>62</td>
</tr>
<tr>
<td>Automobile Rental</td>
<td>50</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>Meals</td>
<td>125</td>
<td>0</td>
<td>125</td>
</tr>
</tbody>
</table>
### List Units Hours Quantity

<table>
<thead>
<tr>
<th>List</th>
<th>Units</th>
<th>Hours</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other Asset</td>
<td>160</td>
<td>0</td>
<td>160</td>
</tr>
<tr>
<td>Other Expenses</td>
<td>225</td>
<td>0</td>
<td>225</td>
</tr>
<tr>
<td>Lodging</td>
<td>330</td>
<td>0</td>
<td>330</td>
</tr>
<tr>
<td>Other Expenses</td>
<td>225</td>
<td>0</td>
<td>225</td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA_ACCUM_CMT_TXNS_V</td>
<td>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.</td>
</tr>
<tr>
<td>PA_ACCUM_RSRC_ACT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACCUM_RSRC_CMT_V</td>
<td>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</td>
</tr>
<tr>
<td>View</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_ACCUM_RSRC_COST_BGT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACCUM_RSRC_REV_BGT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACCUM_WBS_ACT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACCUM_WBS_CMT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACCUM_WBS_COST_BGT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACCUM_WBS_REV_BGT_V</td>
<td>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</td>
</tr>
<tr>
<td>PA_ACT_BY_GL_PERIOD_V</td>
<td>Returns actual cost and revenue totals for lowest tasks and resources by GL periods</td>
</tr>
<tr>
<td>PA_ACT_BY_PA_PERIOD_V</td>
<td>Returns actual cost and revenue totals for lowest tasks and resources by PA periods</td>
</tr>
<tr>
<td>PA_BURDEN_COMPONENT_CMT_V</td>
<td>Returns commitment burden components by resource, PA period name, expenditure type, expenditure organization, and burden set for each transaction summarization record</td>
</tr>
<tr>
<td>View</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PA_BURDEN_COMPONENT_COST_V</td>
<td>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.</td>
</tr>
<tr>
<td>PA_CMT_BY_GL_PERIOD_V</td>
<td>Returns current commitment totals for lowest tasks and resources by GL period.</td>
</tr>
<tr>
<td>PA_CMT_BY_PA_PERIOD_V</td>
<td>Returns current commitment totals for lowest tasks and resources by PA periods.</td>
</tr>
<tr>
<td>PA_GL_PERIODS_V</td>
<td>A view of the PA_PERIODS tables for GL periods and their start and end dates.</td>
</tr>
<tr>
<td>PA_PA_PERIODS_V</td>
<td>A view of the PA_PERIODS tables for PA periods and their start and end dates.</td>
</tr>
<tr>
<td>PA_PM_REFERENCE_V</td>
<td>Retrieves Oracle Projects identifiers and reference codes from your external systems for projects and tasks.</td>
</tr>
<tr>
<td>PA_TASK_ASSIGNMENTS_AMG_V</td>
<td>Retrieves information about all valid task assignment progress for the organization associated with the user's responsibility.</td>
</tr>
<tr>
<td>PA_TASK_PROGRESS_AMG_V</td>
<td>Retrieves information about all valid task progress for the organization associated with the user's responsibility.</td>
</tr>
<tr>
<td>PA_TXN_ACCUM_V</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Status API Procedures**

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

- Update Procedures
  - UPDATE_EARNED_VALUE, page 6-83
  - UPDATE_PROGRESS, page 6-84

- Load-Execute-Fetch Procedures
  - LOAD_TASK_PROGRESS, page 6-86
  - EXECUTE_UPDATE_TASK_PROGRESS, page 6-86
  - INIT_UPDATE_TASK_PROGRESS, page 6-87

Record and Table Datatypes
The record and table datatype used in the APIs is defined on the following pages.

**PA_TASK_PROGRESS_LIST_REC_TYPE Datatype**
The following table shows the PA_TASK_PROGRESS_LIST_REC_TYPE datatype.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TASK_ID</td>
<td>NUMBER</td>
<td>Yes, if PM_TASK_REFERENCE is not provided</td>
<td>The reference code that uniquely identifies the task within a project in Oracle Projects.</td>
</tr>
<tr>
<td>TASK_NAME</td>
<td>VARCHAR2 (20)</td>
<td>No</td>
<td>Task name</td>
</tr>
<tr>
<td>TASK_NUMBER</td>
<td>VARCHAR2 (25)</td>
<td>No</td>
<td>The element number or task number of the task</td>
</tr>
<tr>
<td>PM_TASK_REFERENCE</td>
<td>VARCHAR2 (250)</td>
<td>Yes, if TASK_ID is not provided</td>
<td>The reference code that uniquely identifies the structure or task in the external system.</td>
</tr>
<tr>
<td>PERCENT_COMPLETE</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The percent complete of the object</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2 (250)</td>
<td>No</td>
<td>The progress overview or description</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The project element ID of the structure, task, or deliverable, or the resource list member ID of the assignment in Oracle Projects</td>
</tr>
<tr>
<td>OBJECT_VERSION_ID</td>
<td>NUMBER</td>
<td>Yes</td>
<td>The element version ID of the task or deliverable</td>
</tr>
<tr>
<td>OBJECT_TYPE</td>
<td>VARCHAR2 (30)</td>
<td>Yes</td>
<td>The object type: PA_STRUCTURES for project level progress; PA_TASKS for task level progress; PA_ASSIGNMENTS for assignment level progress; PA_DELIVERABLES for deliverable level progress</td>
</tr>
<tr>
<td>PROGRESS_STATUS_CODE</td>
<td>VARCHAR2 (150)</td>
<td>Yes</td>
<td>Progress status code</td>
</tr>
<tr>
<td>PROGRESS_COMMENT</td>
<td>VARCHAR2 (4000)</td>
<td>No</td>
<td>Progress comments</td>
</tr>
<tr>
<td>ACTUAL_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Actual start date</td>
</tr>
<tr>
<td>ACTUAL_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Actual finish date</td>
</tr>
<tr>
<td>ESTIMATED_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Estimated start date</td>
</tr>
<tr>
<td>ESTIMATED_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Estimated finish date</td>
</tr>
<tr>
<td>SCHEDULED_START_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Scheduled start date</td>
</tr>
<tr>
<td>SCHEDULED_FINISH_DATE</td>
<td>DATE</td>
<td>No</td>
<td>Scheduled finish date</td>
</tr>
<tr>
<td>TASK_STATUS</td>
<td>VARCHAR2 (150)</td>
<td>Yes</td>
<td>The task status code or deliverable status code</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Required?</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------</td>
<td>-----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>EST_REMAINING_EFFORT</td>
<td>NUMBER</td>
<td>No</td>
<td>The estimate to complete effort</td>
</tr>
<tr>
<td>ACTUAL_WORK_QUANTITY</td>
<td>NUMBER</td>
<td>No</td>
<td>Actual work quantity; applies only at the lowest level tasks</td>
</tr>
<tr>
<td>LOWEST_LEVEL_TASK</td>
<td>VARCHAR2 (1)</td>
<td>For internal use only</td>
<td></td>
</tr>
<tr>
<td>PROGRESS_MODE</td>
<td>VARCHAR2 (30)</td>
<td>For internal use only</td>
<td></td>
</tr>
<tr>
<td>ETC_COST</td>
<td>NUMBER</td>
<td>No</td>
<td>Estimate to complete cost</td>
</tr>
<tr>
<td>PM_DELIVERABLE_REFERENCE</td>
<td>VARCHAR2 (150)</td>
<td>No</td>
<td>Deliverable reference</td>
</tr>
<tr>
<td>PM_TASK_ASSIGN_REFERENCE</td>
<td>VARCHAR2 (150)</td>
<td>No</td>
<td>Task assignment reference</td>
</tr>
<tr>
<td>ACTUAL_COST_TO_DATE</td>
<td>NUMBER</td>
<td>No</td>
<td>The cumulative actual cost in transaction currency</td>
</tr>
<tr>
<td>ACTUAL_EFFORT_TO_DATE</td>
<td>NUMBER</td>
<td>No</td>
<td>The cumulative actual effort</td>
</tr>
</tbody>
</table>

### Required Parameters and Parameter Values

The following tables shows the parameters that are required when you use the UPDATE_PROGRESS and LOAD_TASK PROGRESS APIs to update progress information for projects, tasks, assignments, and deliverables.

### Parameters Required for Projects

The following table shows the parameters that are required when you use the UPDATE_PROGRESS and LOAD_TASK PROGRESS APIs to update progress information for projects.
<table>
<thead>
<tr>
<th><strong>Parameter Name</strong></th>
<th><strong>Comment</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PROJECT_ID</td>
<td>Required if P_PM_PROJECT_REFERENCE is not provided</td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
<tr>
<td>P_AS_OF_DATE</td>
<td></td>
</tr>
<tr>
<td>P_TASK_ID</td>
<td>Value = 0</td>
</tr>
<tr>
<td>P_PM_TASK_REFERENCE</td>
<td>Required if P_TASK_ID is not provided</td>
</tr>
<tr>
<td>P_OBJECT_ID</td>
<td>Value = STRUCTURE ID</td>
</tr>
<tr>
<td>P_OBJECT_VERSION_ID</td>
<td>Value = STRUCTURE VERSION ID</td>
</tr>
<tr>
<td>P_OBJECT_TYPE</td>
<td>Value = PA_STRUCTURES</td>
</tr>
<tr>
<td>P_PROGRESS_STATUS_CODE</td>
<td></td>
</tr>
<tr>
<td>P_TASK_STATUS</td>
<td></td>
</tr>
</tbody>
</table>

**Parameters Required for Tasks**

The following table shows the parameters that are required when you use the UPDATE_PROGRESS and LOAD_TASK PROGRESS APIs to update progress information for tasks.

<table>
<thead>
<tr>
<th><strong>Parameter Name</strong></th>
<th><strong>Comment</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PROJECT_ID</td>
<td>Required if P_PM_PROJECT_REFERENCE is not provided</td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Comment</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
<tr>
<td>P_AS_OF_DATE</td>
<td></td>
</tr>
<tr>
<td>P_TASK_ID</td>
<td>Required if P_PM_TASK_REFERENCE is not provided</td>
</tr>
<tr>
<td>P_PM_TASK_REFERENCE</td>
<td>Required if P_TASK_ID is not provided</td>
</tr>
<tr>
<td>P_OBJECT_ID</td>
<td>Value = TASK ID</td>
</tr>
<tr>
<td>P_OBJECT_VERSION_ID</td>
<td>Value = TASK VERSION ID</td>
</tr>
<tr>
<td>P_OBJECT_TYPE</td>
<td>Value = PA_TASKS</td>
</tr>
<tr>
<td>P_PROGRESS_STATUS_CODE</td>
<td></td>
</tr>
<tr>
<td>P_TASK_STATUS</td>
<td></td>
</tr>
</tbody>
</table>

### Parameters Required for Assignments

The following table shows the parameters that are required when you use the UPDATE_PROGRESS and LOAD_TASK_PROGRESS APIs to update progress information for assignments.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PROJECT_ID</td>
<td>Required if P_PM_PROJECT_REFERENCE is not provided</td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
</tbody>
</table>
The following table shows the parameters that are required when you use the UPDATE_PROGRESS and LOAD_TASK_PROGRESS APIs to update progress information for deliverables.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PROJECT_ID</td>
<td>Required if P_PM_PROJECT_REFERENCE is not provided</td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
<tr>
<td>P_AS_OF_DATE</td>
<td></td>
</tr>
</tbody>
</table>

**Parameters Required for Deliverables**

The following table shows the parameters that are required when you use the UPDATE_PROGRESS and LOAD_TASK_PROGRESS APIs to update progress information for deliverables.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_PROJECT_ID</td>
<td>Required if P_PM_PROJECT_REFERENCE is not provided</td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
<tr>
<td>P_PM_PROJECT_REFERENCE</td>
<td>Required if P_PM_PROJECT_ID is not provided</td>
</tr>
<tr>
<td>P_PM_PRODUCT_CODE</td>
<td></td>
</tr>
<tr>
<td>P_PM_STRUCTURE_TYPE</td>
<td></td>
</tr>
<tr>
<td>P_AS_OF_DATE</td>
<td></td>
</tr>
</tbody>
</table>
### Parameter Name | Comment
--- | ---
P_TASK_ID | Required if P_PM_TASK_REFERENCE is not provided
P_PM_TASK_REFERENCE | Required if P_TASK_ID is not provided
P_OBJECT_ID | Value = DELIVERABLE ID
P_OBJECT_VERSION_ID | Value = DELIVERABLE VERSION ID
P_OBJECT_TYPE | Value = PA_DELIVERABLES

### Related Topics
- UPDATE_PROGRESS, page 6-84
- LOAD_TASK_PROGRESS, page 6-86

### Status API Procedure Definitions
This section contains descriptions of the status APIs.

#### 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.

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
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for UPDATE_EARNED_VALUE are listed below:

- P_API_VERSION_NUMBER
- P_PROJECT_ID
- P_TASK_ID
- P_RESOURCE_LIST_MEMBER_ID
- P_AS_OF_DATE
- P_BCWS_ITD
- P_ACWP_ITD
- P_BCWP_ITD
- P_BAC_ITD
- P_BQWS_ITD
- P_AQWP_ITD
- P_BQWP_ITD
- P_BAQ_ITD

**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 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.

**Note:** See "Status API Parameters", page 6-79, for additional information about whether parameters are required when updating progress information for projects, tasks, assignments, or deliverables.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for UPDATE_PROGRESS are listed below:

- **P_API_VERSION_NUMBER**
- **P_PROJECT_ID** (if P_PM_PROJECT_REFERENCE is not provided)
- **P_PM_PROJECT_REFERENCE** (if P_PM_PROJECT_ID is not provided)
- **P_PM_PRODUCT_CODE**
- **P_PM_STRUCTURE_TYPE**
- **P_AS_OF_DATE**
- **P_TASK_ID** (if P_PM_TASK_REFERENCE is not provided)
- **P_PM_TASK_REFERENCE** (if P_TASK_ID is not provided)
- **P_PERCENT_COMPLETE**
- **P_OBJECT_ID**
- **P_OBJECT_VERSION_ID**
- **P_OBJECT_TYPE**
- **P_PROGRESS_STATUS_CODE**
• **P_TASK_STATUS**

**LOAD_TASK_PROGRESS**

LOAD_TASK_PROGRESS is a Load-Execute-Fetch PL/SQL procedure used to load progress information in the pl/sql data structures.

**Note:** See "Status API Parameters", page 6-79, for additional information about whether parameters are required when updating progress information for projects, tasks, assignments, or deliverables.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for LOAD_TASK_PROGRESS are listed below:

• **P_API_VERSION_NUMBER**

• **P_PROJECT_ID** (if **P_PM_PROJECTREFERENCE** is not provided)

• **P_PM_PROJECTREFERENCE** (if **P_PM_PROJECTID** is not provided)

• **P_PM_PRODUCT_CODE**

• **P_PM_STRUCTURE_TYPE**

• **P_AS_OF_DATE**

• **P_TASK_ID** (if **P_PM_TASKREFERENCE** is not provided)

• **P_PM_TASKREFERENCE** (if **P_TASKID** is not provided)

• **P_PERCENT_COMPLETE**

• **P_OBJECT_ID**

• **P_OBJECT_VERSION_ID**

• **P_OBJECT_TYPE**

• **P_PROGRESS_STATUS_CODE**

• **P_TASK_STATUS**

**EXECUTE_UPDATE_TASK_PROGRESS**

EXECUTE_UPDATE_TASK_PROGRESS is a Load-Execute-Fetch PL/SQL procedure used to update progress information in Oracle Projects.
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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for EXECUTE_UPDATE_TASK_PROGRESS are listed below:

- P_API_VERSION_NUMBER

**INIT_UPDATE_TASK_PROGRESS**

INIT_UPDATE_TASK_PROGRESS is a utility used to initialize internal PL/SQL tables. It is recommended to call this API before every call to the EXECUTE_UPDATE_TASK_PROGRESS API.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for INIT_UPDATE_TASK_PROGRESS are listed below:

- P_API_VERSION_NUMBER

**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 retrieve amounts for a single period or a range of periods for either Oracle Projects or Oracle General Ledger periods. You can retrieve actual cost, revenue, and commitment amounts.

To obtain an understanding of how this API can be used for reporting, refer to the descriptions of the following Oracle Projects reports in the *Oracle Projects Fundamentals* guide:

- Revenue, Cost, Budgets by Resources (Project Level)
• Task - Revenue, Cost, Budgets by Resources

The Actuals Summarization API package name is `pa_accum_api`. The corresponding file names are as follows:

• Body file: PAAAPIB.pls

• Specification file: PAAAPIS.pls

The Actuals Summarization API includes the following procedures:

• GET_PROJECT_ACCUM_ACTUALS, page 6-88

• GET_PROJ_RES_ACCUM, page 6-88

• GET_PROJ_TXN_ACCUM, page 6-89

**Actuals Summarization API Procedures**

You can view descriptions of all of the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**GET_PROJECT_ACCUM_ACTUALS**

This is the primary procedure for retrieving actual cost, revenue, and commitment amounts at a project level.

The required parameters for GET_PROJECT_ACCUM_ACTUALS are listed below:

• `x_project_id`

• `x_from_period_name`

*Important:* If a value is passed to this procedure for the parameter `x_resource_list_member_id` while retrieving aggregate project or task amounts, then the procedure calls the GET_PROJ_RES_ACCUM procedure. If no value is passed for the parameter `x_resource_list_member_id`, then this procedure calls the GET_PROJ_TXN_ACCUM procedure.

**GET_PROJ_RES_ACCUM**

This procedure retrieves actual cost, revenue, and commitment amounts for the resource named in the parameter `x_resource_list_member_id` for the procedure GET_PROJECT_ACCUM_ACTUALS.

The required parameters for this procedure are listed below:

• `x_project_id`
• x_from_period_name
• x_resource_list_member_id

GET_PROJ_TXN_ACCUM

This procedure retrieves summarized amounts for the various transaction types such as cost, revenue and commitment.

The required parameters for this procedure are listed below:
• x_project_id
• x_from_period_name

Budget Summarization API

You can use the Budget Summarization API for custom reporting. This API gets budget data for any budget baseline. You can get the budget data without running the Update Project Summary process.

Note: The Budget Summarization API does not support budgets created using financial plan types.

This 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 Summarization 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. For example, if a project is budgeted at the top task and you pass a lower task to the Budget Summarization API, the 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.
Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Project Performance Reporting APIs

The Project Performance Reporting APIs enable you to summarize base summary data for resources across projects without navigating to the resource summary or resource analysis pages for each project. You can then analyze the resource summary amounts for multiple projects to create your own custom reports. When you no longer require the data, Oracle Projects recommends that you use the Project Performance Reporting APIs to delete the resource rollup summary amounts that you created and improve system performance.

The Project Performance Reporting API package name is pji_perf_rptg_pub. The corresponding file names are as follows:

- Body file: PJIPRFPB.pls
- Specification file: PJIPRFPSP.ls

The Project Performance Reporting API includes the following procedures:

- CREATE_RESOURCE_ROLLUP, page 6-90
- DELETE_RESOURCE_ROLLUP, page 6-91

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

CREATE_RESOURCE_ROLLUP

This procedure inserts header information for the specified projects and structures in the PJI_ROLLUP_LEVEL_STATUS table and the corresponding rollup resource summary amount details in the PJI_FP_XBS_ACCUM_F table.

The following CREATE_RESOURCE_ROLLUP parameters are required:

- P_API_VERSION_NUMBER
- P_COMMIT
- P_INIT_MSG_LIST
- P_PROJECT_ID
• P_PLAN_VERSION_ID_TBL
• P_RBS_VERSION_ID_TBL
• P_PRG_ROLLUP_FLAG
• X_MSG_COUNT
• X_MSG_DATA
• X_RETURN_STATUS

DELETE_RESOURCE_ROLLUP
This procedure deletes header information for the specified projects and structures from the PJI_ROLLUP_LEVEL_STATUS table and the corresponding rollup resource summary amount details from the PJI_FP_XBS_ACCUM_F table.

The following DELETE_RESOURCE_ROLLUP parameters are required:
• P_API_VERSION_NUMBER
• P_COMMIT
• P_INIT_MSG_LIST
• P_PROJECT_ID
• X_MSG_COUNT
• X_MSG_DATA
• X_RETURN_STATUS
Part 3

ORACLE PROJECTS CLIENT EXTENSIONS
Overview of Client Extensions

This chapter describes everything you need to know about designing and writing client extensions in Oracle Projects.

This chapter covers the following topics:

- Client Extensions
- Implementing Client Extensions

Client Extensions

You can 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.

The package specification and body (template procedure) files are stored in the Oracle Projects patch/115/sql directory. 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. 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.

Related Topics

Analyzing Your Business Requirements, page 7-2
Designing the Logic, page 7-2
Writing PL/SQL Procedures, page 7-4

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 of Business Requirements Analysis

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.

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 7-2.)

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:

```plaintext
IF charging supplies
    THEN IF charging to overhead projects
        THEN OK
        ELSE error message
    ELSE OK
You can charge supplies only to overhead projects
```

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:

```plaintext
IF Expenditure Type = Supplies
    THEN IF Project Type = Overhead
        THEN OK
        ELSE error message
    ELSE OK
You can charge supplies only to overhead projects
```

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 below:

**Package Specification File**

The specification file is the interface to your applications. It declares the types, variables, constants, exceptions, cursors,
Overview of Client Extensions

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.

**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.

**Tip:** 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...
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:

```sql
PROCEDURE name [ (parameter [, parameter,...]) ] IS
    [local declarations]
BEGIN
    executable statements
    [EXCEPTION
        exception handlers]
END [name];
```

The parameter syntax above expands to the following syntax:

```sql
var_name [IN | OUT NOCOPY | IN OUT NOCOPY] datatype [{:= | DEFAULT} value]
```


Using Template Procedures

Oracle Projects provides 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 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):

```bash
$ 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:

```bash
$ 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.
This chapter describes the client extensions in the Oracle Project Foundation application.

This chapter covers the following topics:

- Project Security Extension
- Project Verification Extension
- Project and Task Date Extension
- Project Workflow Extension
- Verify Organization Change Extension
- Transaction Import Extensions
- Descriptive Flexfield Mapping Extension
- Archive Project Validation Extension
- Archive Custom Tables Extension

**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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPSECXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPSECXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_security_extn</td>
</tr>
<tr>
<td>Procedure</td>
<td>check_project_access</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

You can view descriptions of the parameters for CHECK_PROJECT_ACCESS in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAXBUEBU</td>
<td>Budgets window</td>
</tr>
<tr>
<td>PAXCARVW</td>
<td>Capital Projects window</td>
</tr>
<tr>
<td>PAXINEAG</td>
<td>Agreements window</td>
</tr>
<tr>
<td>PAXINRVW</td>
<td>Invoice Review window</td>
</tr>
<tr>
<td>PAXINVPF</td>
<td>Project Funding Inquiry window</td>
</tr>
<tr>
<td>PAXPREPR</td>
<td>Projects window</td>
</tr>
<tr>
<td>PAXTRAPE.PROJECT</td>
<td>Project Expenditure Inquiry window</td>
</tr>
<tr>
<td>PAXURVPS</td>
<td>Project Status Inquiry window</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXPCECB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXPCECS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_proj_status</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

Verify Project Status Change

The name for this procedure is 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Check Workflow Enabled

The name for this procedure is 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Project and Task Date 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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPMGCEB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPMGCES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_pm</td>
</tr>
<tr>
<td>Procedure</td>
<td>customize_dates</td>
</tr>
</tbody>
</table>
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.

**Important:** 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 7-4.

### Customize Dates

The name of the customize dates procedure is `pa_client_extn_pm.customize_dates`.

You can view descriptions of all of the parameters for this procedure in the Oracle
Integration Repository. The Oracle Integration Repository is described in the preface of
this manual.

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
-- whose data you import,
-- 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;
ELSEIF p_pm_product_code = {different product code} THEN
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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAWFPCEB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAWFPCES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_project_wf</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

Procedures

Following are the procedures included in the project workflow extension.

Select Project Approver

The name of this procedures is 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.
Start Project Workflow

The name of this procedure is start_project_wf.

This procedure starts the workflow process for project status changes.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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.

Location and Package Name

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXORCEB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXORCES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_org_client_extn</td>
</tr>
<tr>
<td>Procedure</td>
<td>verify_org_change</td>
</tr>
</tbody>
</table>

Important: 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
Verify Organization Change

The name for the verify organization change extension is verify_org_change. This procedure is described below.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Related Topics

Mass Update Batches, Oracle Projects Fundamentals
Function Security in Oracle Projects, Oracle Projects Implementation Guide

Transaction Import 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
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.

**Note:** Oracle Projects does not support the use of a Pre-Import client extension or a Post-Import client extension with the *Capitalized Interest* transaction source.

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 8-11
- Post-Import Client Extension for Internet Time, page 8-13

You may refer to the existing client extensions for Internet Time when you create additional Pre-Import and Post-Import client extensions.

**Related Topics**

- Transaction Import Interface, page 13-25
- 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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXTTRXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXTTRXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_trx_import</td>
</tr>
<tr>
<td>Procedure</td>
<td>pre_import</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Pre-Import Procedure Parameters**

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of
this manual.

Related Topics

Transaction Import Client Extensions, page 8-10
Transaction Import Interface, page 13-25
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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAXTTRXS.pls</td>
</tr>
<tr>
<td>Body template</td>
<td>PAXTTRXB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_trx_import</td>
</tr>
<tr>
<td>Procedures</td>
<td>post_import</td>
</tr>
</tbody>
</table>
Post-Import Procedure Parameters

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Related Topics

Transaction Import Client Extensions, page 8-10
Transaction Import Interface, page 13-25
Transaction Sources, Oracle Projects Implementation Guide
Transaction Source Options, Oracle Projects Implementation Guide

Descriptive Flexfield Mapping Extension

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: Oracle Projects holds 10 descriptive flexfield segments. If you are using more than 10 segments in Payables, only the first 10 are imported to Oracle Projects.

Description

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAPDFCS.pls</td>
</tr>
<tr>
<td>Body template</td>
<td>PAPDFCCB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_dfftrans</td>
</tr>
</tbody>
</table>
### Important:
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 7-4.

### Arguments Passed by the Calling Modules

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)

### PRC: Interface Expense Reports from Payables (PAAPIMP)

The following table shows the arguments that the Interface Expense Reports from Payables process passes to the client extension:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_TRX_REF_1</td>
<td>AP_INVOICE_DISTRIBUTIONS.INVOICE_ID</td>
</tr>
<tr>
<td>P_TRX_REF_2</td>
<td>AP_INVOICE_DISTRIBUTIONS.DISTRIBUTION_LINE_NUMBER</td>
</tr>
<tr>
<td>P_TRX_TYPE</td>
<td>AP_INVOICES.INVOICE_TYPE_LOOKUP_CODE</td>
</tr>
<tr>
<td>P_SYSTEM_LINKAGE_FUNCTION</td>
<td>&quot;VI&quot;</td>
</tr>
<tr>
<td>P_SUBMODULE</td>
<td>AP_INVOICES.SOURCE</td>
</tr>
</tbody>
</table>
PRC: Interface Supplier Invoices from Payables (PAAPIMP)

The following table shows the arguments that the Interface Supplier Invoices from Payables process passes to the client extension:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_TRX_REF_1</td>
<td>AP_INVOICE_DISTRIBUTIONS.INVOICE_ID</td>
</tr>
<tr>
<td>P_TRX_REF_2</td>
<td>AP_INVOICE_DISTRIBUTIONS.DISTRIBUTION_LINE_NUMBER</td>
</tr>
<tr>
<td>P_TRX_TYPE</td>
<td>&quot;EXPENSE REPORT&quot;</td>
</tr>
<tr>
<td>P_SYSTEM_LINKAGE_FUNCTION</td>
<td>&quot;ER&quot;</td>
</tr>
<tr>
<td>P_SUBMODULE</td>
<td>AP_INVOICES.SOURCE</td>
</tr>
</tbody>
</table>

PRC: Interface Expense Reports to Payables (PATTER)

The following table shows the arguments that the Interface Expense Reports to Payables process passes to the client extension:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_TRX_REF_1</td>
<td>PA_EXPENDITURE_ITEMS.EXPENDITURE_ITEM_ID</td>
</tr>
<tr>
<td>P_TRX_REF_2</td>
<td>PA_COST_DISTRIBUTION_LINES.LINE_NUM</td>
</tr>
<tr>
<td>P_TRX_TYPE</td>
<td>&quot;EXPENSE REPORT&quot;</td>
</tr>
<tr>
<td>P_SYSTEM_LINKAGE_FUNCTION</td>
<td>&quot;ER&quot;</td>
</tr>
<tr>
<td>P_SUBMODULE</td>
<td>null</td>
</tr>
</tbody>
</table>

PRC: Interface Supplier Invoice Adjustment Costs to Payables (PAVTVC)

The following table shows the arguments that the Interface Supplier Invoice Adjustment Costs to Payables process passes to the client extension:
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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_TRX_REF_1</td>
<td>PA_EXPENDITURE_ITEMS.EXPENDITURE_ITEM_ID</td>
</tr>
<tr>
<td>P_TRX_REF_2</td>
<td>PA_COST_DISTRIBUTION_LINES.LINE_NUM</td>
</tr>
<tr>
<td>P_TRX_TYPE</td>
<td>AP_INVOICES.INVOICE_TYPE_LOOKUP_CODE</td>
</tr>
<tr>
<td>P_SYSTEM_LINKAGE_FUNCTION</td>
<td>&quot;VI&quot;</td>
</tr>
<tr>
<td>P_SUBMODULE</td>
<td>null</td>
</tr>
</tbody>
</table>
Related Topics

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.

Description

This extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXAPVXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXAPVXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Purge_Extn.Validate</td>
</tr>
<tr>
<td>Procedure</td>
<td>Validate_Extn</td>
</tr>
</tbody>
</table>

Important: 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 7-4.

Validate Projects Procedure

The name for this procedure is Validate_Extn. By default, the procedure returns NULL to the calling program.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXAPPXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXAPPXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Purge_Extn</td>
</tr>
<tr>
<td>Procedure</td>
<td>PA_Purge_Client_Extn</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

### 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.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
This chapter describes the client extensions in the Oracle Project Costing application.

This chapter covers the following topics:

- Transaction Control Extensions
- AutoApproval Extensions
- Labor Costing Extensions
- Labor Transaction Extensions
- Overtime Calculation Extension
- Burden Costing Extension
- Burden Resource Extension
- Allocation Extensions
- Asset Allocation Basis Extension
- Asset Assignment Extension
- Asset Lines Processing Extension
- Capital Event Processing Extension
- Capitalized Interest Extension
- CIP Grouping Extension
- CIP Account Override Extension
- Depreciation Account Override Extension
- Cross-Charge Client Extensions

**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

**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?

**Writing Transaction Control Extensions**

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXTTCXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXTTCXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>Patcx</td>
</tr>
<tr>
<td>Procedure</td>
<td>tc_extension</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**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.

**Warning:** During a software upgrade, messages are not preserved.
Before you upgrade Oracle Projects to a new release, be sure that you move the messages to a custom application and set the parameter *Delete customer added rows* to NO. After the upgrade, move the messages from the custom application to Oracle Projects.

The messages are stored in the table FND_NEW_MESSAGES.


**Procedure**

The name of the transaction control extension procedure is tc_extension.

Parameter values for this procedure are passed from the expenditure item that is being validated. You can view descriptions of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Additional Information About Parameters**

Following is additional information about the parameters for this procedure.

- **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 Modules

The calling module parameter indicates which program called the transaction control extension. You can vary the logic of the extension based 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 are 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.

The following list shows possible values for X_CALLING_MODULE, and the corresponding meaning for each value:

- **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.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateRelated Item</td>
<td>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.</td>
</tr>
<tr>
<td>PAVVIT</td>
<td>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.</td>
</tr>
<tr>
<td>PAXTREPE</td>
<td>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.</td>
</tr>
<tr>
<td>PAXTRTRX</td>
<td>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.</td>
</tr>
<tr>
<td>PAXEXCOP/ PAXEXCB</td>
<td>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.</td>
</tr>
<tr>
<td>PAXPRRPE</td>
<td>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.</td>
</tr>
<tr>
<td>PAXVOTCB</td>
<td>Oracle Time and Labor</td>
</tr>
<tr>
<td>PAXVSSTS</td>
<td>Oracle Internet Time</td>
</tr>
<tr>
<td>POWERBREQ</td>
<td>iProcurement</td>
</tr>
<tr>
<td>POXPOEPO</td>
<td>Purchase Orders in Oracle Purchasing. This value is passed when Transaction Controls is called to validate project-related information entered on a purchase order.</td>
</tr>
<tr>
<td>POXRQERQ</td>
<td>Requisitions in Oracle Purchasing. This value is passed when Transaction Controls is called to validate project-related information entered on a requisition.</td>
</tr>
</tbody>
</table>
**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.

**Project Deliverables**
This value is passed when Transaction Controls is called to validate expenditure information for a deliverable procurement action.

**REQIMPORT**
Requisition import

**SelfService**
Expense reports

**Frequently Asked Questions**
Following are frequently asked questions regarding the transaction control extension.

**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 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 the extension validate groups of expenditure items?**
Currently, you cannot perform validation on groups of expenditure items.

**How many error messages can the procedure return?**
Your procedure can return one error message, which is the first error message that Oracle Projects encounters in your procedure.

**Related Topics**
Designing Client Extensions, page 7-2

**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.

*Tip:* 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.

Tip: 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:

<table>
<thead>
<tr>
<th>Determining Incurred by Organization</th>
<th>Because 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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determining Task Organization</td>
<td>Task organization is not passed as a parameter to the transaction control extension. Therefore, you need to derive this value.</td>
</tr>
<tr>
<td>Determining Task Service Type</td>
<td>The task service type is not passed as a parameter to the transaction control extension. Therefore, you need to derive this value.</td>
</tr>
<tr>
<td>Defining an Error Message</td>
<td>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, &quot;Only the task-owning organization can charge to this task&quot;.</td>
</tr>
</tbody>
</table>

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.

**Tip:** Review the sample PL/SQL code that corresponds to the implementation of this case study, view the file PAXTTTCXB.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:

<table>
<thead>
<tr>
<th>Deriving Additional Information</th>
<th>Because 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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determining Billable Status</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 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.

You can view descriptions of all of the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Important:** 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 7-4.

The AutoApproval extensions include the following extensions:

**AutoApproval Profile Options Extension**

This procedure performs custom validation for all the expenditure items in an expenditure. The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Template</td>
<td>PAXPTEEB.pls</td>
</tr>
<tr>
<td>Specification Template</td>
<td>PAXPTEES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_pte</td>
</tr>
<tr>
<td>Procedure</td>
<td>get_exp_autoapproval</td>
</tr>
</tbody>
</table>

**Expenditure Summary AutoApproval Extension**

This extension contains default logic to read the values of the AutoApproval profile options. The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Template</td>
<td>PAXTGTCB.pls</td>
</tr>
<tr>
<td>Specification Template</td>
<td>PAXTGTCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pagtcx</td>
</tr>
<tr>
<td>Procedure</td>
<td>summary_validation_extension</td>
</tr>
</tbody>
</table>

**Timecard AutoApproval Extension**

Use this procedure to incorporate additional approval logic for timecards. The extension is identified by the following items:
AutoApproval Routing Extension

Use this procedure to define rules for routing timecards and expense reports for approval. The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Template</td>
<td>PAXTRTEB.pls</td>
</tr>
<tr>
<td>Specification Template</td>
<td>PAXTRTES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>paroutingx</td>
</tr>
<tr>
<td>Procedure</td>
<td>route_to_extension</td>
</tr>
</tbody>
</table>

Timecard Entry AutoApproval Extension

Use this procedure to define validations during entry and approval of timecards in Oracle Time and Labor. For more information about this client extension, see the Oracle Time and Labor Implementation and User Guide.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Template</td>
<td>PAPSSTCB.pls</td>
</tr>
<tr>
<td>Specification Template</td>
<td>PAPSSTCS.pls</td>
</tr>
</tbody>
</table>
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
Writing Labor Costing Extensions

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXCCECB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXCCECS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Client_Extn_Costing</td>
</tr>
<tr>
<td>Procedure</td>
<td>Calc_Raw_Cost</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

Calculate Raw Cost

The name of this procedure is PA_Client_Extn_Costing.Calc_Raw_Cost.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Using x_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 x_status**

Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

\[
x_{status} = 0 \quad \text{The extension executed successfully.}
\]
An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.

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.

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?

**Writing Labor Transaction Extensions**

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXCCETB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXCCETS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Client_Extn_Txn</td>
</tr>
<tr>
<td>Procedure</td>
<td>Add_Transactions</td>
</tr>
</tbody>
</table>

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

**Important:** 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 7-4.

**Add Related 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.
You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Create Related Transactions

Use the `pa_transactions.CreateRelatedItem` 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.

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

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
Updating Related Transactions

Use the `pa_transactions.UpdateRelatedItem` 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 is located in the `pa_transactions` package. You cannot change this procedure.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Information About Parameters

Details about some of the parameters are 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_DISTRIBUTION_REJECTION_CODE and
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.

**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.

**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.

### Related Topics

Distributing Labor Costs, *Oracle Project Costing User Guide*
Overtime Calculation Extension

The overtime calculation extension enables 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 9-16.

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.

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.

Location and Package Name

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXDLCOB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXDLCOS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_calc_overtime</td>
</tr>
</tbody>
</table>

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: Location and Package Name, page 9-26.

The following illustration 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.
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.

Related Topics

Overview of Tracking Overtime, Oracle Project Costing User Guide

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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAXCCEBS.pls</td>
</tr>
<tr>
<td>Body template</td>
<td>PAXCCEBB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_burden</td>
</tr>
<tr>
<td>Procedure</td>
<td>override_rate_rev_id</td>
</tr>
</tbody>
</table>

Important: 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 7-4.
Override Burden Schedule ID

The Override Burden Schedule ID procedure (override_rate_rev_id) assigns a burden cost schedule to a transaction.

You can view descriptions of all of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Related Topics

Entering Project and Task Options, Oracle Projects Fundamentals
Rate Schedules, Oracle Projects Implementation Guide

Burden Resource Extension

You can use the burden resource extension to control the reporting of burden transactions so that summary burden transactions and their source raw transactions are reported under the same planning resource in a resource breakdown structure.

This extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXBRGCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXBRGCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CLIENT_EXTN_BURDEN_RESOURCE</td>
</tr>
<tr>
<td>Function</td>
<td>CLIENT_GROUPING</td>
</tr>
<tr>
<td>Procedure</td>
<td>CLIENT_COLUMN_VALUES</td>
</tr>
</tbody>
</table>

Client Grouping

The CLIENT_GROUPING function returns a VARCHAR2 value which is a concatenated string of the parameter values. You can customize the function to create the return string using the attributes by which you want to group each transaction. This string can be used as an additional grouping criterion.
Client Column Values

The CLIENT_COLUMN_VALUES procedure works in conjunction with the CLIENT_GROUPING function. It returns NULL for the parameters that are not used for additional grouping in the CLIENT_GROUPING function.

You can view descriptions of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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 9-30
- Allocation Target Extension, page 9-32
- Allocation Offset Tasks Extension, page 9-34
- Allocation Offset Projects and Tasks Extension, page 9-35
- Allocation Basis Extension, page 9-36
- Allocation Descriptive Flexfields Extension, page 9-37
- Allocation Dependencies Extension, page 9-38

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPALCCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPALCCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>source_extn</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Business Rules**

The following business rules apply to this extension.

- The source project and the allocation rule must be from the same operating unit.
- The source task must be a top or lowest task.

**Parameters**

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Datatypes**

The datatype `alloc_source_tabtype` contains the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT_ID</td>
<td>NUMBER</td>
<td>(Required) Identifier of the source project.</td>
</tr>
<tr>
<td>TASK_ID</td>
<td>NUMBER</td>
<td>Identifier of the source task</td>
</tr>
</tbody>
</table>
Parameter | Type | Description
--- | --- | ---
EXCLUDE_FLAG | VARCHAR2(1) | (Default is N) Exclusion flag. If the value is Y, the project and task are excluded from the source project and tasks.

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPALCCB.pls</td>
</tr>
</tbody>
</table>
### Item Name

<table>
<thead>
<tr>
<th>Specification template</th>
<th>PAPALCCS.pls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>target_extn</td>
</tr>
</tbody>
</table>

### Procedure Parameters

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

### Datatype Parameters

One of the parameters for the allocation target extension is the datatype `alloc_target_tabtype`. This datatype contains the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT_ID</td>
<td>NUMBER</td>
<td>(Required) Identifies the target project. If cross-charging is enabled, target projects and source projects can be in different operating units.</td>
</tr>
<tr>
<td>TASK_ID</td>
<td>NUMBER</td>
<td>Identifies the target task (task must be chargeable)</td>
</tr>
<tr>
<td>PERCENT</td>
<td>NUMBER</td>
<td>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 9-33.</td>
</tr>
<tr>
<td>EXCLUDE_FLAG</td>
<td>VARCHAR2(1)</td>
<td>(Default is N) If Y, exclude the project and task from the target project and tasks</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPALCCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPALCCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>offset_task_extn</td>
</tr>
</tbody>
</table>

Parameters

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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 returns 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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Template</td>
<td>PAPALCCB.pls</td>
</tr>
<tr>
<td>Specification Template</td>
<td>PAPALCCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>offset_extn</td>
</tr>
</tbody>
</table>

**Procedure Parameters**

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Datatype Parameters**

One of the parameters for the allocation target extension is the datatype alloc_offset_tabtype. This datatype 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)

**Validation**

The Generate Allocation Transactions process:

- Validates the project_id against the single organization view pa_projects
- Validates 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
- Validates 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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPALCCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPALCCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>basis_extn</td>
</tr>
</tbody>
</table>

Procedure Parameters

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPALCCB.pls</td>
</tr>
</tbody>
</table>
### Item Name

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAPALCCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>txn_dff_extn</td>
</tr>
</tbody>
</table>

#### Procedure Parameters

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

#### 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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPALCCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPALCCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>check_dependency</td>
</tr>
</tbody>
</table>

#### Parameters

The extension uses the following parameters:
### Parameter Usage Type Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Usage</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_ALLOC_RULE_ID</td>
<td>IN</td>
<td>NUMBER</td>
<td>(Required) Identifies the allocation rule</td>
</tr>
<tr>
<td>X_STATUS</td>
<td>OUT</td>
<td>NUMBER</td>
<td>(Required) Indicates if an error occurred:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOCOPY</td>
<td>=0 Successful validation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&lt;0 Oracle error; message is written to a log file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&gt;0 Application error</td>
</tr>
<tr>
<td>X_ERROR_MESSAGE</td>
<td>OUT</td>
<td>VARCHAR2(30)</td>
<td>Error message text</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOCOPY</td>
<td></td>
</tr>
</tbody>
</table>

### Related Topics

Allocations, *Oracle Project Costing User Guide*

### Asset Allocation Basis Extension

This extension enables you to define your own allocation bases for allocating unassigned and common costs across multiple project assets.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCXAAB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACCXAAS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_asset_alloc</td>
</tr>
<tr>
<td>Procedure</td>
<td>asset_alloc_basis</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.
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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Related Topics

- Implementing Client Extensions, page 7-2
- Allocating Asset Costs, Oracle Project Costing User Guide

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:
### Important:
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 7-4.

### Parameters
You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

### 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.

### Related Topics
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.

The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPGALCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPGALCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_gen_asset_lines</td>
</tr>
<tr>
<td>Procedure</td>
<td>client_asset_assignment</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPGALCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPGALCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_gen_asset_lines</td>
</tr>
<tr>
<td>Procedure</td>
<td>client_asset_assignment</td>
</tr>
</tbody>
</table>
### 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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

### Related Topics

- Implementing Client Extensions, page 7-2
- Generating Summary Asset Lines, *Oracle Project Costing User Guide*
- Generate Asset Lines, *Oracle Projects Fundamentals*

### 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.

The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCXCBB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACCXCBS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_pre_cap_event</td>
</tr>
<tr>
<td>Procedure</td>
<td>pre_capital_event</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Related Topics**

Implementing Client Extensions, page 7-2
Creating Capital Events, *Oracle Project Costing User Guide*
Create Periodic Capital Events, *Oracle Projects Fundamentals*

**Capitalized Interest Extension**

The capitalized interest client extension enables you to customize the capitalized
interest calculation process.

The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACINTXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACINTXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_cap_int</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Procedures**

The following procedures are provided in the capitalized interest client extension.

You can view the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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.

**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.

**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.
Interest Override Procedure

The procedure name is calculate_cap_interest.

The Interest Override procedure enables you to define your own calculations for capitalized interest.

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.

Grouping Method Procedure

The procedure name is grouping_method. The Grouping Method procedure enables you to specify grouping criteria.

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.

Related Topics

Implementing Client Extensions, page 7-2
Capitalizing Interest, Oracle Project Costing User Guide
Capitalized Interest, Oracle Projects Implementation Guide

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.

The extension is identified by the following items:
<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAXGCES.pls</td>
</tr>
<tr>
<td>Body template</td>
<td>PAXGCEB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_exten_cip_grouping</td>
</tr>
<tr>
<td>Function</td>
<td>client_grouping_method</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Client_Grouping_Method Function**

You can view the parameters for this function in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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_id IN PA_EXPENDITURE_ITEMS_ALL.expenditure_item_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
Related Topics

Project Types: Capitalization Information, *Oracle Projects Implementation Guide*

Creating a Capital Asset in Oracle Projects, *Oracle Project Costing User Guide*

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

The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCXCOB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACCXCOS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_cip_acct_ovr</td>
</tr>
<tr>
<td>Procedure</td>
<td>cip_acct_override</td>
</tr>
</tbody>
</table>

**Important**: 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 7-4.

CIP Account Override Procedure

The procedure name *iscip_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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Related Topics**

Implementing Client Extensions, page 7-2

Creating and Preparing Asset Lines for Oracle Assets, *Oracle Project Costing User Guide*

**Depreciation Account Override Extension**

This extension, enables you to specify logic for deriving the depreciation expense account that is assigned to a project asset. The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCXDEB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACCXDES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_deprn_exp_ovr</td>
</tr>
<tr>
<td>Procedure</td>
<td>deprn_exp_acct_override</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

The procedure name is `isdeprn_exp_acct_override`. Oracle Projects calls this procedure during the update of the Assets and Asset Details windows, and during validation of asset information when you submit the PRC: Interface Assets process.

Before the PRC: Interface Assets 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.

You can view the parameters for this procedure in the Oracle Integration Repository.
The Oracle Integration Repository is described in the preface of this manual.

Related Topics
Implementing Client Extensions, page 7-2
Defining and Processing Assets, Oracle Project Costing User Guide
Interface Assets, Oracle Projects Fundamentals.

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 9-51
Cross Charge Processing Method Override Extension, page 9-52
Transfer Price Determination Extension, page 9-54
Transfer Price Override Extension, page 9-55
Transfer Price Currency Conversion Override Extension, page 9-57
Internal Payables Invoice Attributes Override Extension, page 9-58

Related Topics
Cost Accrual Identification Extension, page 10-34

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCIXTB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAACCIXTS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CC_IDENT_CLIENT_EXTN</td>
</tr>
<tr>
<td>Procedure</td>
<td>override_prvdr_recvr</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

Parameters

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCIXTB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACCIXTS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CC_IDENT_CLIENT_EXTN</td>
</tr>
<tr>
<td>Procedure</td>
<td>override_cc_processing_method</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Prerequisites**

The transaction must be a cross-charged transaction.

**Parameters**

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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)

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 9-55.

Description

This extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPTPRCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPTPRCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CC_TP_CLIENT_EXTN</td>
</tr>
<tr>
<td>Procedure</td>
<td>determine_transfer_price</td>
</tr>
</tbody>
</table>
**Important:** 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 7-4.

**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**

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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
Description

This extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPTPRCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPTPRCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CC_TP_CLIENT_EXTN</td>
</tr>
<tr>
<td>Procedure</td>
<td>override_transfer_price</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

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

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPMCECB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPMCECS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_MULTI_CURR_CLIENT_EXTN</td>
</tr>
<tr>
<td>Procedure</td>
<td>override_curr_conv_attributes</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**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**

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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.

**Description**

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACCINPB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACCINPS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CC_AP_INV_CLIENT_EXTN</td>
</tr>
<tr>
<td>Procedure</td>
<td>override_exp_type_exp_org</td>
</tr>
</tbody>
</table>

**Important:** 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
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

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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.

Related Topics

Defining Provider and Receiver Controls, Oracle Projects Implementation Guide
This chapter describes the client extensions in the Oracle Project Billing application. This chapter covers the following topics:

- Funding Revaluation Factor Extension
- Billing Cycle Extension
- Billing Extensions
- Cost Accrual Billing Extension
- Cost Accrual Identification Extension
- Labor Billing Extensions
- Non-Labor Billing Extensions
- Retention Billing Extension
- Automatic Invoice Approve/Release Extension
- Output Tax Extension
- Receivables Installation Override Extension
- AR Transaction Type Extension

**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.

The extension is identified by the following items:
<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXBFRCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXBFRCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>Pa_Client_Extn_Funding_Reval</td>
</tr>
<tr>
<td>Procedure</td>
<td>Funding_Revaluation_factor</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXIBCXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXIBCXS.pls</td>
</tr>
</tbody>
</table>
### Package \(\text{pa\_client\_extn\_bill\_cycle}\)

### Procedure \(\text{get\_next\_billing\_date}\)

**Important:** 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 7-4.

**Warning:** Do not use the PL/SQL commands Commit and Rollback in your billing extension code. For the \(\text{get\_next\_billing\_date}\) function, define the pragma RESTRICT_REFERENCES as WNDS, WNPS. For more information, refer to the PL/SQL User’s Guide and Reference Manual.

The procedure \(\text{pa\_client\_extn\_bill\_cycle}\).get\_next\_billing\_date returns a value for the next billing date.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

### 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 \(\text{PA\_BILLING\_EXTN\_PARAMS\_V}\). 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 extension.
**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 10-24.

**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 MyPackageName.insert_event to create automatically created events. This is shown in the following table.
Implementing Billing Extensions

The following illustration shows the steps required to implement billing extensions.

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 10-5.

**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.

**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

**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. The following table shows the attributes of Fremont Corporation’s billing extension.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Billing Extension Name</td>
<td>Communication Charge</td>
</tr>
<tr>
<td>Calling Process</td>
<td>Invoice</td>
</tr>
<tr>
<td>Default Event Type</td>
<td>Surcharge</td>
</tr>
<tr>
<td>Default Event Description</td>
<td>Communication Charge</td>
</tr>
<tr>
<td>Check Boxes Checked</td>
<td>Adjustment Processing</td>
</tr>
<tr>
<td></td>
<td>Regular Processing</td>
</tr>
<tr>
<td></td>
<td>Percentage</td>
</tr>
<tr>
<td></td>
<td>Project Specific</td>
</tr>
<tr>
<td>Default Cost Budget Type</td>
<td>Approved Cost Budget</td>
</tr>
<tr>
<td>Default Revenue Budget Type</td>
<td>Approved Revenue Budget</td>
</tr>
</tbody>
</table>

**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 10-11.
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 extensions 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 7-2.

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.

<table>
<thead>
<tr>
<th>Billing Extension</th>
<th>Event</th>
<th>Revenue Amount</th>
<th>Bill Amount</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>Bill amount is processed after the related revenue is distributed</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>100</td>
<td>-</td>
<td>Revenue event only</td>
</tr>
</tbody>
</table>

The following table shows examples of events that you can create in the Generate Draft Invoice calling process.

<table>
<thead>
<tr>
<th>Billing Extension</th>
<th>Event</th>
<th>Revenue Amount</th>
<th>Bill Amount</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>Revenue amount is processed after the related invoice is released</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-</td>
<td>100</td>
<td>Invoice event only</td>
</tr>
</tbody>
</table>

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

• 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

The following list describes each of the calling places.

**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. 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.
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.

<table>
<thead>
<tr>
<th>Period</th>
<th>Invoice Number</th>
<th>Invoice Credited</th>
<th>Invoice Amount</th>
<th>Automatic Event Amount (Transaction Dependent)</th>
<th>Automatic Event Amount (Transaction Independent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>1000</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-500</td>
<td>-50</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>1500</td>
<td>150</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Summary</td>
<td></td>
<td></td>
<td>2000</td>
<td>200</td>
<td>300</td>
</tr>
</tbody>
</table>

**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.
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.

**Tip:** 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.

<table>
<thead>
<tr>
<th>Event Attribute</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Description</td>
<td>Default value is event description on billing extension.</td>
</tr>
</tbody>
</table>
**Event Attribute** | **Comments**
---|---
Event Type | Defaults value is event type on billing extension. Event type classification must be *Automatic*.
Event Organization | Default value is 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.

**Budget Attributes**

When you design 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 default values for any of these attributes.

<table>
<thead>
<tr>
<th>Budget Attribute</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost Budget Type Code</td>
<td>Default value is approved cost budget.</td>
</tr>
<tr>
<td>Revenue Budget Type Code</td>
<td>Default value is approved revenue budget.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXITMPB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXITMPS.pls</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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 the X_top_task_id 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 MyPackageName.

**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 10-22
- Identify transactions processed in the current run, page 10-22
- Insert events, page 10-23
- Retrieve budget amounts, page 10-24
- Handle error conditions, page 10-24

### 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.

The procedure name is `MyPackageName_pub.insert_event`. You can view descriptions of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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*
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.

The procedure name is MyPackageName.get_budget_amount.

You can view descriptions of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The parameters include input and output parameters for cost and revenue budget type codes.

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.

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.

The name of this procedure is MyPackageName.insert_message.

You can view descriptions of the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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.

**Tip:** 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 a 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: Perform 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
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.

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:
  \[ \text{Surcharge} = \text{Surcharge Percentage} \times \text{Labor Amount Invoiced} \]

- 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.
Tip: To review the sample PL/SQL code that corresponds to the implementation of this case study, view the file PAITMPS.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?

\[
\text{Surcharge} = \text{Surcharge Percentage} \times \text{Labor Invoiced}
\]

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.

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, because there is a corporate default percentage that will be used when project users do not enter a negotiated percentage.

**Tip:** 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 of a Surcharge Billing Extension

The following table shows attributes of a sample surcharge billing extension.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Billing Extension Name</td>
<td>Communication Charge</td>
</tr>
<tr>
<td>Procedure</td>
<td>pa_demo_surcharge.execute</td>
</tr>
<tr>
<td>Description</td>
<td>Calculate surcharge to invoice based on percentage of labor invoiced</td>
</tr>
<tr>
<td>Order</td>
<td>20</td>
</tr>
<tr>
<td>Default Event Type</td>
<td>Surcharge</td>
</tr>
<tr>
<td>Default Event Description</td>
<td>Communication Charge</td>
</tr>
<tr>
<td>Calling Place: Revenue</td>
<td>No</td>
</tr>
<tr>
<td>Calling Place: Invoice</td>
<td>Yes</td>
</tr>
<tr>
<td>Calling Place: Preprocessing</td>
<td>No</td>
</tr>
<tr>
<td>Calling Place: Adjustment</td>
<td>Yes</td>
</tr>
<tr>
<td>Calling Place: Regular</td>
<td>Yes</td>
</tr>
<tr>
<td>Calling Place: Post-Processing</td>
<td>No</td>
</tr>
<tr>
<td>Required Input: Amount</td>
<td>No</td>
</tr>
<tr>
<td>Required Input: Percentage</td>
<td>No</td>
</tr>
<tr>
<td>Product-Specific</td>
<td>Yes</td>
</tr>
<tr>
<td>Transaction Independent</td>
<td>No</td>
</tr>
</tbody>
</table>

### Testing

You specify the following test cases to use in testing your billing extension procedure.
You now have all of the components of your functional design to give to your technical resource for writing the PL/SQL procedures.

**Related Topics**

- Designing Client Extensions, page 7-2
- Event Types, *Oracle Projects Implementation Guide*
- Defining Project Types, *Oracle Projects Implementation Guide*

**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.

The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXICOSB.pls</td>
</tr>
<tr>
<td>Item</td>
<td>Name</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXICOSS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_REV_CA</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

You can view the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

For more information about using the Cost Accrual Extension, see Revenue-Based Cost Accrual, *Oracle Project Billing User Guide*.

**Calculation Procedure**

The calculation procedure (calc_ca_amt) is the main procedure for calculating and generating the cost accrual entries.

**PSI Cost Accrual Procedure**

The PSI cost accrual procedure (get_psi_cols) displays the cost accrual columns in Project Status Inquiry.

**Verify Project Status for Cost Accrual Procedure**

The name for this procedure is *verify_project_status_ca*. This procedure is called when a user changes a project’s status.

**Check Cost Accrual Procedure**

The name for this procedure is *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.

**Cost Accrual Identification Extension**

Use this extension to identify cross charged projects that use cost accrual during

The extension includes the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAICPCAB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAICPCAS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_CC_CA</td>
</tr>
<tr>
<td>Procedure</td>
<td>identify_ca_projects</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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

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:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXICTMB.pls</td>
</tr>
<tr>
<td>Item</td>
<td>Name</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXICTMS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Client_Extn_Billing</td>
</tr>
<tr>
<td>Procedure</td>
<td>Calc_Bill_Amount</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Business Rules**

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.

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?

**Calculate Bill Amount**

The procedure name is `PA_Client_Extn_Billing.Calc_Bill_Amount`.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Following is additional information about parameters for this procedure.
<table>
<thead>
<tr>
<th>Amount</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• No value is passed in to the x_amount parameter. Do not expect an amount in this parameter when you create calculations in the extension.</td>
<td></td>
</tr>
<tr>
<td>• The client extension must assign a value to the parameter x_amount, or else the extension will be ignored by the calling program.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bill Rate</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>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:</td>
<td></td>
</tr>
<tr>
<td>• B (specifies bill rate)</td>
<td></td>
</tr>
<tr>
<td>• null or value other than B (specifies markup)</td>
<td></td>
</tr>
<tr>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Markup Percentage</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>If you specify that your amount is based on markup, you should populate x_markup_percentage with the markup percentage amount, so that the expenditure item record will contain accurate data.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the x_status parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:</td>
<td></td>
</tr>
<tr>
<td>x_status = 0</td>
<td>The extension executed successfully.</td>
</tr>
<tr>
<td>x_status &lt; 0</td>
<td>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.</td>
</tr>
<tr>
<td>x_status &gt; 0</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Related Topics**

Revenue Flow, Oracle Project Billing User Guide
Non-Labor Billing Extensions

Non-labor billing extensions enable you to derive labor billing amounts for individual non-labor transactions. You can use non-labor billing extensions to implement unique non-labor billing methods. Some examples of non-labor billing extensions you can define are:

- Tiered pricing method
- External system rate derivation

The Non Labor Billing Extension is called during the revenue generation process to determine non-labor revenue and billing amounts. The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXINCTB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXINCTS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_non_labor_bill_clt_extn</td>
</tr>
<tr>
<td>Procedure</td>
<td>calc_bill_amount</td>
</tr>
</tbody>
</table>

**Important:** 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.

Business Rules

Oracle Projects processes non-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 non-labor billing transaction, it skips the standard bill amount and rate calculation section of the revenue process for that transaction.

Consider the following design issues for non-labor billing extensions:

- What are the conditions and circumstances under 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 require? How are the rates and other information maintained?
• What are the exception conditions for your non-labor billing extension? What is the exception handling if you cannot find a rate that should exist?

Calculate Bill Amount

The procedure name is Pa_Non_Labor_Bill_Clt_Extn.Calc_Bill_Amount.

Information About Parameters

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Following is additional information about parameters for this procedure.

• **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_DISTR_REJ_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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXBRTCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXBRTCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_retention</td>
</tr>
<tr>
<td>Procedure</td>
<td>BILL_RETENTION</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Related Topics

Retention Billing, *Oracle Project Billing User Guide*

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.

The extension is identified by the following items:
### Important
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 7-4.

### Business Rules
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.

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?

### Information About Parameters
You can view the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. Following is additional information about parameters for this client extension.

**x_invoice_class**

The valid values of x_invoice_class are:
### Value Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVOICE</td>
<td>regular invoice</td>
</tr>
<tr>
<td>CREDIT_MEMO</td>
<td>crediting invoice</td>
</tr>
<tr>
<td>WRITE_OFF</td>
<td>write-off invoice</td>
</tr>
<tr>
<td>CANCEL</td>
<td>canceling invoice</td>
</tr>
</tbody>
</table>

#### x_status

Use the `x_status` parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x_status = 0</code></td>
<td>The extension executed successfully.</td>
</tr>
<tr>
<td><code>x_status &lt; 0</code></td>
<td>An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.</td>
</tr>
<tr>
<td><code>x_status &gt; 0</code></td>
<td>An application error occurred. Oracle Projects writes a rejection reason to the PA_DISTRIBUTION_WARNINGS table. The invoice is not approved or released.</td>
</tr>
</tbody>
</table>

### Output Tax Extension

You set up a hierarchy for Oracle Projects and the project operating unit in the application tax options of Oracle E-Business Tax. The Generate Draft Invoices process uses the Application Tax Options hierarchy to determine the default tax classification codes on invoice lines. The Output Tax client extension is one of the default tax options in the Application Tax Options hierarchy.

The Generate Draft Invoices process calls the Output Tax extension if it does not find the default tax classification code from the other tax options you defined in the Application Tax Options hierarchy. You can use the extension to satisfy your business rules in assigning the default tax classification code for invoice lines.

The extension is identified by the following items:
<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAXPOTXS.pls</td>
</tr>
<tr>
<td>Body template</td>
<td>PAXPOTXB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_output_tax</td>
</tr>
<tr>
<td>Procedure</td>
<td>get_tax_codes</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

The name for this procedure is get_tax_codes. The get tax codes procedure assigns a tax classification code to an invoice line.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Related Topics**

Setting Up Tax for Oracle Project Invoices in Oracle E-Business Tax, *Oracle Projects Implementation Guide*

*Oracle E-Business Tax User Guide*

**Receivables Installation Override Extension**

The Receivables Installation Override client extension enables 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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification template</td>
<td>PAPARICS.pls</td>
</tr>
<tr>
<td>Item</td>
<td>Name</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Body template</td>
<td>PAPARICB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_override_ar_inst</td>
</tr>
<tr>
<td>Procedure</td>
<td>get_installation_mode</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**Business Rules**

To use this extension, 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.
Get Installation Mode

The get_installation_mode procedure returns an installation mode to the calling program.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Modifying the Get Installation Mode Procedure

The default procedure includes the following PL/SQL statement:
```plsql
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:
```plsql
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.

Oracle Projects calls the AR Transaction Type Extension during the Transfer Invoices to Oracle Receivables process.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXPTRXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXPTRXS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_inv_transfer</td>
</tr>
<tr>
<td>Procedure</td>
<td>get_ar_trx_type</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.
Get AR Transaction Type

The name for this procedure is `pa_client_extn_inv_transfer.get_ar_trx_type`.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Additional Information About Parameters

Following is additional information about the parameters for this extension.

**x_invoice_class**

The valid values of `x_invoice_class` are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVOICE</td>
<td>regular invoice</td>
</tr>
<tr>
<td>CREDIT_MEMO</td>
<td>crediting invoice</td>
</tr>
<tr>
<td>WRITE_OFF</td>
<td>write-off invoice</td>
</tr>
<tr>
<td>CANCEL</td>
<td>canceling invoice</td>
</tr>
</tbody>
</table>

**x_status**

Use the `x_status` parameter to handle error conditions for your procedure. This parameter indicates the processing status of your extension as follows:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x_status = 0</code></td>
<td>The extension executed successfully.</td>
</tr>
<tr>
<td><code>x_status &lt; 0</code></td>
<td>An Oracle error occurred and the process did not complete. Oracle Projects writes an error message to the process log file.</td>
</tr>
<tr>
<td><code>x_status &gt; 0</code></td>
<td>An application error occurred. Oracle Projects writes a rejection reason to the PA_DISTRIBUTION_WARNINGS table. The invoice is not approved or released.</td>
</tr>
</tbody>
</table>
This chapter describes the client extensions in the Oracle Project Resource Management application.

This chapter covers the following topics:

- Assignment Approval Changes Extension
- Assignment Approval Notification Extension
- Candidate Notification Workflow Extension

**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.

- Change in transfer price rate override values
  
  A change in the transfer price rate override, transfer price currency override, transfer price basis override, and transfer price applied percent override requires an approval, because it changes the transfer price of the resource.

The default project assignment approval workflow process calls the assignment approval changes extension.
The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PARAAPCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PARAAPCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_asgmt_apprvl</td>
</tr>
<tr>
<td>Function</td>
<td>is_asgmt_appr_items_changed</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

**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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Related Topics**

Implementing Client Extensions, page 7-2

**Assignment Approval Notification Extension**

You can use this client extension to customize the list of default contacts (recipients) used by the assignment approval workflow.

The default project assignment approval workflow process calls the assignment approval notification extension.

The extension is identified by the following items:
Important: 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 7-4.

You can view the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PARAWFCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PARAWFCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_asgmt_wf</td>
</tr>
</tbody>
</table>
• Assignment Cancellation Notification

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Related Topics**

Implementing Client Extensions, page 7-2

**Candidate Notification Workflow Extension**

You can use this client extension to customize the list of people who receive a notification from the PA: Candidate Notification Process Workflow.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PARCWFCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PARCWFCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_cand_wf</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

The USERS_LIST_TBLTYP parameters for this package are shown in the following table:
### Parameter Names, Required Data Types, and Descriptions

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Required</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_NAME</td>
<td>Yes</td>
<td>VARCHAR2</td>
<td>The workflow user name of the approver</td>
</tr>
<tr>
<td>PERSON_ID</td>
<td>Yes</td>
<td>NUMBER</td>
<td>The person ID of the approver</td>
</tr>
<tr>
<td>TYPE</td>
<td>Yes</td>
<td>VARCHAR2</td>
<td>The type of user, such as RESOURCE_MANAGER or PRIMARY_CONTACT</td>
</tr>
<tr>
<td>ROUTING_ORDER</td>
<td>No</td>
<td>NUMBER</td>
<td>The order in which the approvals should be submitted. (For FYI notification recipients, this value is ignored because such notifications are sent to all recipients at the same time.)</td>
</tr>
</tbody>
</table>

### 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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

### Related Topics

Implementing Client Extensions, page 7-2
This chapter describes the client extensions in the Oracle Project Management application.

This chapter covers the following topics:
- Workplan Workflow Extension
- Progress Management Extension
- Budget Calculation Extensions
- Budget Verification Extension
- Budget Workflow Extension
- Estimate to Complete Generation Method Extension
- Control Item Document Numbering Extension
- Issue and Change Workflow Extension
- Project Status Report Workflow Extension
- Custom Performance Measure Extension
- Project Performance Status Extension
- Project Status Inquiry (PSI) Extension
- Project Status Inquiry Burdening Commitments Extension
- Project Status Inquiry Commitment Changes Extension

**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

The default workplan workflow process calls the workplan workflow extension. The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXSTWCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXSTWCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_workplan_workflow_client</td>
</tr>
</tbody>
</table>

You can view the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

**Start Workflow Procedure**

The procedure name is `start_workflow`.

This procedure starts the workflow process for a workplan.

**Select Approver Procedure**

The procedure name is `select_approver`.

This procedure determines the approver for the workplan approval process.

**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.

**Related Topics**

Designing Client Extensions, page 7-2

**Progress Management Extension**

The Progress Management client extension overrides the default method of deriving actual and estimated dates for lowest tasks and task assignments at any structure level. The extension is identified by the following items:
The name of the procedure is `get_task_res_override_info`.

This procedure overrides the default method to calculate actual and estimated dates for lowest level tasks and task assignments at all workplan structure levels. It runs for all lowest tasks and task assignments whenever you:

- Submit progress for a lowest task
- Submit progress for a task assignment
- Run the summarization process

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

## 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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPCTCXS.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPCTCXB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_progress_client_extn</td>
</tr>
<tr>
<td>Procedure</td>
<td>get_task_res_override_info</td>
</tr>
</tbody>
</table>

---

Oracle Project Management Client Extensions  12-3
<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXBCECB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXBCECS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Client_Extn_Budget</td>
</tr>
</tbody>
</table>

The names of the procedures are:
- calc_raw_cost
- calc_burdened_cost
- calc_revenue

**Business Rules**

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?

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

**Procedures**

Following are descriptions of the procedures for this client extension.

You can view the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
Calculate Raw Cost

The name for this procedure is \textit{calc\_raw\_cost}.

Calculate Burdened Cost

The name for this procedure is \textit{calc\_burdened\_cost}.

\textbf{Tip:} 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 4-7.

Calculate Revenue

The name for this procedure is \textit{calc\_revenue}.

Additional Information About Parameters

Following is additional information about the parameters for this client extension.

\textbf{Error Handling}

Use the \texttt{x\_error\_code}, \texttt{x\_error\_message}, \texttt{p\_error\_code}, and \texttt{p\_error\_message} parameters to help resolve error conditions should your procedure fail.

The \texttt{x\_err\_code} or \texttt{p\_error\_code} parameter indicates the processing status of your procedure as follows:

\textbf{Tip:} 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.

\begin{itemize}
  \item \texttt{x\_error\_code = 0} \quad \text{The procedure executed successfully.}
  \item \texttt{x\_error\_code < 0} \quad \text{An Oracle error occurred and the process did not complete.}
  \item \texttt{x\_error\_code > 0} \quad \text{An application error occurred and the}
\end{itemize}
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>.

Related Topics

Designing Client Extensions, page 7-2

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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXBCECB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXBCECS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PA_Client_Extn_Budget</td>
</tr>
<tr>
<td>Procedures</td>
<td>verify_budget_rules, stamp_client_extn_errors</td>
</tr>
</tbody>
</table>

You can view the parameters for these procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Verify Budget Rules

The procedure name is 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.

**Stamp Client Extension Errors**

The procedure name is `stamp_client_extn_errors`. The system uses it to display error messages when you upload a project workplan from an Excel spreadsheet. When you perform this action, the system calls both the Budget Calculation Extension and the Budget Verification extension.

For this procedure to work correctly, you must define lookup codes for all of the error messages that you would like to apply to an uploaded budget.

**Related Topics**

Designing Client Extensions, page 7-2

**Budget Workflow Extension**

The budget workflow extension enables you to customize the workflow processes for changing the status of a budget or forecast.

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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAWFBCEB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAWFBCES.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_budget_wf</td>
</tr>
</tbody>
</table>

Before you implement this extension, you must define the rules that will determine whether to call Oracle Workflow to baseline a budget or forecast, and to select the budget or forecast approver.

You can view the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.
Determine Whether to Call Workflow

The name of this procedure is `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.

Start the Budget Workflow

The name of this procedure is `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.

Specify Budget Verification Rules

The name of this procedure is `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`.

Determine the Approver

The name of this procedure is `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.

Estimate to Complete Generation Method Extension

The Estimate to Complete Generation Method extension enables you to control the calculation of estimate to complete (ETC) quantities and amounts in forecasts.

You can use this extension to calculate quantities and amounts for raw cost, burdened cost, and revenue.

The following items identify the extension:
ETC Calculation Procedure

The name of this procedure is `fcst_gen_client_ext`.

Use this procedure to define calculations for ETC quantities and amounts for raw cost, burdened cost, and revenue.

Following is the PL/SQL table record type definition:

- type `l_pds_rate_dtls_rec_type` is RECORD
  
  ( PERIOD_NAME pa_budget_lines.period_name%TYPE, RAW_COST_RATE pa_budget_lines.txn_standard_cost_rate%TYPE, BURDENED_COST_RATE pa_budget_lines.burden_cost_rate%TYPE, REVENUE_BILL_RATE pa_budget_lines.txn_standard_bill_rate%TYPE);

- type `l_pds_rate_dtls_tab` is

  TABLE 1 of `l_pds_rate_dtls_rec_type`;

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACINRXB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACINRXS.pls</td>
</tr>
</tbody>
</table>
Get Next Number

The name of the procedure 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.

You can view the parameters for this procedure in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Related Topics

Implementing Client Extensions, page 7-2
Control Item Types, *Oracle Projects Implementation Guide*

Issue and Change Workflow Extension

This extension enables you to customize the workflow processes for submitting and approving issues and change documents.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACIWFCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACIWFCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_control_items_wf_client</td>
</tr>
</tbody>
</table>

You can view the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Start Workflow

The name of this procedure is `start_workflow`. Use this procedure to start the workflow.
process for issue and change document approval.

Set Control Item Approver

The name of this procedure is set_ci_approver. Use this procedure to specify persons that can approve issues and change documents.

Set Notification Party

The name of this procedure is set_notification_party. Use this procedure to specify persons to notify for approved and rejected issues and change documents.

Related Topics

Implementing Client Extensions, page 7-2
Contol Item Types, Oracle Projects Implementation Guide

Project Status Report Workflow Extension

The project status report workflow extension enables you to customize the workflow processes for approving and publishing a project status report. You must determine how you want to approve and publish the report. See Overview of Project Status Reports, Oracle Project Management User Guide.

The default project status report workflow process calls the project status report workflow extension.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPRWFCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPRWFCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_report_workflow_client</td>
</tr>
</tbody>
</table>

You can view the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

Start Workflow

The name of this procedure is start_workflow. You can use this workflow to change or
add workflow parameters and to start the workflow.

**Status Report Approver**

The name of this procedure 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 status report. The default procedure sends a notification to the approver that you specified for the report on the Status Report Setup Details page. If you did not specify an approver on this page, then the supervisor of the person who reported the report becomes the approver.

**Notification Party**

The name of this procedure is `set_report_notification_party`. This procedure determines which users receive workflow notifications when a project status report is approved, published, or rejected.

**Related Topics**

Designing Client Extensions, page 7-2

**Custom Performance Measure Extension**

This client extension enables you to create custom measures for reporting project performance.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PJISCO1B.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PJISCO1S.pls</td>
</tr>
<tr>
<td>Package</td>
<td>PJI_PJP_SUM_CUST</td>
</tr>
</tbody>
</table>

You can view the procedure parameters in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

The Custom Performance Measure client extension provides the following procedures:

**Planning Custom Measure Procedure**

The procedure name is `PJP_CUSTOM_FPR_API`.
Use this procedure to insert a maximum of 15 custom measures for financial and plan amounts into the columns labeled CUSTOM1- CUSTOM15 of the PJJ_FP_CUST_PJPO table.

Project performance reporting summarization processes consider these new custom measures and inserts performance data lines against the measures in the PJJ_FP_XBS_ACCUM_F table.

**Activity Custom Measure Procedure**

The procedure name is PJP_CUSTOM_ACR_API.

Use this procedure to insert a maximum of 15 custom measures for bookings and invoice amounts into the columns labeled CUSTOM1- CUSTOM15 of the PJJ_FP_CUST_PJPO table.

Project performance reporting summarization processes consider these new custom measures and inserts performance data lines against the measures in the PJJ_AC_XBS_ACCUM_F table.

**Related Topics**

Implementing Client Extensions, page 7-2

Project Performance Reporting, *Oracle Projects Implementation Guide*

**Project Performance Status Extension**

The Project Performance Status extension enables you to customize the logic involved in retrieving the overall performance status of the project. This extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAPESCLB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAPESCLS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_perf_status_client_extn</td>
</tr>
<tr>
<td>Procedure</td>
<td>get_performance_status</td>
</tr>
</tbody>
</table>

Use the get_performance_status procedure to derive and retrieve the overall performance status for the project. You can write code to define the following relationships:
• the relationship between the severity and the numeric value given in the PREDEFINED_FLAG column of the PA_LOOKUPS view

• the relationship between the key performance area statuses and the overall performance status of the project

You can view the procedure parameters in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual. The required parameters for this procedure are listed below:

• P_OBJECT_TYPE

• P_OBJECT_ID

• P_KPA_SUMMARY

Related Topics

Implementing Client Extensions, page 7-2
Control Item Types, Oracle Projects Implementation Guide
Project Performance Tracking, Oracle Projects Implementation Guide

Project Status Inquiry (PSI) Extension

You can use a project status inquiry (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 extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXVPS2B.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXVPS2S.pls</td>
</tr>
</tbody>
</table>
You can view the parameters for the following procedures in the Oracle Integration Repository. The Oracle Integration Repository is described in the preface of this manual.

### 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.

### The PSI Get Totals Procedure

The PSI Get Totals procedure consists of two functions for PSI Project window totals functionality:

- Hide_Totals
- Proj_Tot.CustomButton

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 NOCOPY-parameters. The Project window displays NULL for any OUT NOCOPY-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.

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.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Null</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT_ID</td>
<td>NOT NULL</td>
<td>NUMBER(15)</td>
</tr>
<tr>
<td>COLUMN1</td>
<td>NOT NULL</td>
<td>VARCHAR2(240)</td>
</tr>
<tr>
<td>COLUMN2</td>
<td>NOT NULL</td>
<td>VARCHAR2(240)</td>
</tr>
<tr>
<td>COLUMN3</td>
<td>NOT NULL</td>
<td>VARCHAR2(240)</td>
</tr>
<tr>
<td>COLUMN4 - COLUMN33</td>
<td>NOT NULL</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

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;

Related Topics

Project Summary Amounts, Oracle Project Management User Guide

Project Status Inquiry Burdening Commitments Extension

You can use the Project Status Inquiry Burdening Commitments client extension to control the display of burden cost amounts on commitments.

For a project type, you can choose burdening options that enable you to either store burden amounts on the same expenditure item with raw cost amounts or store burden amounts as separate expenditure items. This extension enables you to override the burdening options defined for projects types for purposes of displaying burden amounts for commitment transactions in Project Status Inquiry.

You can use the Burdening Commitments client extension to override the setup for display of burden costs on commitments.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PAXBSGCB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PAXBSGCS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_burden_summary</td>
</tr>
</tbody>
</table>

The name of the function is Same_Line_Burden_Cmt.

This default value returned by the function is False. When the value is set to False, Oracle Projects displays burden amounts on commitments based on the burdening option you choose for displaying burden amounts for a project type.

If you specify a value of True, Oracle Projects always displays burden amounts for
commitment transactions on the same expenditure item with raw cost amounts, regardless of the burdening option you choose for a project type.

**Related Topics**

(All Project Types) Costing Information, *Oracle Projects Implementation Guide*

Drilling Down to Actuals, Commitments, and Events Detail, *Oracle Project Management User Guide*

Storing, Accounting, and Viewing Burden Costs, *Oracle Project Costing User Guide*

**Project Status Inquiry 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.

**Important:** 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.

The extension is identified by the following items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body template</td>
<td>PACECMTB.pls</td>
</tr>
<tr>
<td>Specification template</td>
<td>PACECMTS.pls</td>
</tr>
<tr>
<td>Package</td>
<td>pa_client_extn_check_cmt</td>
</tr>
<tr>
<td>Procedure</td>
<td>commitments_changed</td>
</tr>
</tbody>
</table>

**Important:** 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 7-4.

The name of the procedure is 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 since 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.
Part 4

ORACLE PROJECTS OPEN INTERFACES
This chapter describes the open interfaces in the Oracle Projects applications.

This chapter covers the following topics:

- Transaction Import
- Transaction Import Interface

### 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.

Related Topics

Transaction Import Interface, page 13-25
Expenditure Item Validation, Oracle Project Costing User Guide
Integrating with Oracle Project Manufacturing, Oracle Projects Fundamentals

Transaction Import Process Diagram

The following illustration shows the flow of the transaction import process.
The Transaction Import process flow includes the following steps:

1. Load transactions from external cost collection systems into the PA Transaction Interface Table.

2. Select all pending transactions that satisfy the selection criteria of the process report and validate each transaction.

3. If there are invalid transactions, then the process flow includes the following steps:
1. Update rejected transactions in the interface table using the Review Transactions window (depending on the transaction source). These records appear in the PA Transaction Interface Audit Table (automatically updated with original and adjusting transactions) and the PA Transaction Interface Table.

2. After deleting rejected transactions from the interface table, fix the appropriate data in the external source and re-import the transactions.

3. Return to Step 1.

4. If all transactions are valid, then create expenditure records in Oracle Projects.

5. If the transaction source is purgeable, then delete transactions from the interface table. If not, then update the status of the transactions in the interface table to Accepted.

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
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.

Uploading Expenditure Batches from Microsoft Excel

You can enter and upload pre-approved expenditure batches using Microsoft Excel spreadsheets. The upload process populates the transaction import table. You can optionally use the upload parameter to run the transaction import process automatically.

For additional information, see: Uploading Expenditure Batches from Microsoft Excel, Oracle Project Costing User Guide.

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.

**Important:** Do not use predefined transaction sources for supplier costs and expense reports, Oracle Labor Distribution, project allocations, and capitalized interest transactions when you run the PRC: Transaction Import program. For additional information about predefined transaction sources, see: Transaction Sources, Oracle Projects
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 13-53

Output Reports
Transaction Import has two output reports:
• an exception report, which lists all rejected transactions
• a summary report of successfully imported transactions
Related Topics

Submitting Requests, Oracle Projects Fundamentals
Transaction Import Interface, page 13-25
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.

- Straight Time
- Overtime
- 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 resource 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 8-10

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.

**Related Topics**

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, by 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.

**Note:** You can load items with the expenditure type classes of *Expense Reports* and *Supplier Invoice* only as *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 external systems have already determined the ledger currency raw cost amounts and posted the GL accounts to Oracle General Ledger or Oracle Subledger Accounting. Processes within Oracle Projects do not generate accounting events for these items to send the accounting information to Oracle Subledger Accounting. As a result, Oracle Subledger Accounting does not create accounting for these items. 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.

**Important:** If you import items with both the *GL Accounted* and *Allow Adjustments* options enabled, you can adjust the expenditure items in Oracle Projects. You need to reconcile costs both between Oracle Projects and the external system,
and between Oracle General Ledger and another general ledger application. You may also need to reconcile costs between the external system and Oracle Subledger Accounting.

**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 Oracle Project Manufacturing integration:

<table>
<thead>
<tr>
<th>Transaction Source</th>
<th>Default Expenditure Type Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory</td>
<td>Inventory</td>
</tr>
<tr>
<td>Inventory Misc</td>
<td>Inventory</td>
</tr>
<tr>
<td>Transaction Source</td>
<td>Default Expenditure Type Class</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Inventory with Accounts</td>
<td>Inventory</td>
</tr>
<tr>
<td>Inventory with No Accounts</td>
<td>Inventory</td>
</tr>
<tr>
<td>Work in Process</td>
<td>Work in Process</td>
</tr>
<tr>
<td>WIP with Accounts</td>
<td>Work in Process</td>
</tr>
<tr>
<td>WIP with No Accounts</td>
<td>Work in Process</td>
</tr>
<tr>
<td>WIP Straight Time with Accounts</td>
<td>Straight Time</td>
</tr>
<tr>
<td>WIP Straight Time with No Accounts</td>
<td>Straight Time</td>
</tr>
</tbody>
</table>

**Note:** When a transaction source indicates with No Accounts, Oracle Projects derives accounts for imported transactions using AutoAccounting, generates accounting events, and creates accounting in Oracle Subledger Accounting. For all other transaction sources listed in the preceding table, Oracle Manufacturing derives accounts and creates accounting in Oracle Subledger Accounting, and Oracle Projects imports the accounts. For more information about transaction sources, see: Transaction Sources, Oracle Projects Implementation Guide.

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:

- Because they are transferred to Oracle Projects by sub-element (which maps to the expenditure type), multiple manufacturing transactions associated with one of the predefined the transaction sources can use the same original system reference.

- You can only adjust manufacturing transactions with a transaction source Inventory Misc. in Oracle Projects. You cannot adjust any other manufacturing costs in Oracle Projects, because all accounting for the costs is performed in Oracle Manufacturing. Any adjustments to these costs must originate in Oracle Manufacturing.
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 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 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. Because 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:

<table>
<thead>
<tr>
<th>Functional Currency Rate Type and Rate Date</th>
<th>Project Currency Rate Type and Rate Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following hierarchy is used:</td>
<td>The functional currency attributes are used.</td>
</tr>
<tr>
<td>1. User-entered value</td>
<td></td>
</tr>
<tr>
<td>2. Default value from the lowest task</td>
<td></td>
</tr>
<tr>
<td>3. Default value from the project</td>
<td></td>
</tr>
<tr>
<td>4. Default value from the expenditure</td>
<td></td>
</tr>
<tr>
<td>operating unit's implementation options</td>
<td></td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Functional Currency Rate Type and Rate Date</th>
<th>Project Currency Rate Type and Rate Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following hierarchy is used:</td>
<td>The following hierarchy is used:</td>
</tr>
<tr>
<td>1. User-entered value</td>
<td>1. User-entered value</td>
</tr>
<tr>
<td>2. Default value from the expenditure</td>
<td>2. Default value from the lowest task</td>
</tr>
<tr>
<td>operating unit’s implementation options</td>
<td>3. Default value from the project</td>
</tr>
<tr>
<td></td>
<td>4. For the rate type, the default value from the project operating unit’s implementation options.</td>
</tr>
<tr>
<td></td>
<td>For the rate date, the default value from the expenditure operating unit’s implementation options.</td>
</tr>
</tbody>
</table>

**Rounding Limit for Accounted Multi-Currency Transactions**

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:
<table>
<thead>
<tr>
<th>Column or Calculation</th>
<th>Example 1: Values Within Rounding Limit (transaction is accepted)</th>
<th>Example 2: Values Outside the Rounding Limit (transaction is rejected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction raw cost (DENOM_RAW_COST)</td>
<td>80 GBP</td>
<td>80 GBP</td>
</tr>
<tr>
<td>Functional raw cost (ACCT_RAW_COST)</td>
<td>100 USD</td>
<td>85 USD</td>
</tr>
<tr>
<td>Functional Exchange Rate (based on supplied currency attributes)</td>
<td>1.2375</td>
<td>1.2375</td>
</tr>
<tr>
<td>Rounding Limit (ACCT_EXCHANGE_ROUNDING_LIMIT)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Calculated functional raw cost (DENOM_RAW_COST * Functional Exchange Rate)</td>
<td>99 USD</td>
<td>99 USD</td>
</tr>
<tr>
<td>Difference between calculated and supplied functional raw cost</td>
<td>abs (100 - 99) = 1</td>
<td>abs (85 - 99) = 14</td>
</tr>
</tbody>
</table>

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 Oracle Projects calculates raw cost amounts
- whether Oracle Projects calculates burden amounts
- whether Oracle Projects interfaces amounts to Oracle Payables
- whether Oracle Projects imports the expenditure organization for employee transactions
- whether to allow interface modifications 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 Oracle Projects generates accounting events to create accounting in Oracle Subledger Accounting.

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.

**Important:** 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, Overtime, and Expense Reports**

If the transaction source of the transactions being processed is defined with an expenditure type class of *Straight Time*, *Overtime*, 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, Overtime, 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.

Related Topics

Transaction Import Example: Labor and Expense by Employee Number, page 13-19
Transaction Import Example: Usage, page 13-21

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.

<table>
<thead>
<tr>
<th>Trx Number</th>
<th>Trx Source</th>
<th>Expenditure Type Class</th>
<th>Batch Name</th>
<th>Employee Number</th>
<th>Expenditure Ending Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Site1</td>
<td>Straight Time</td>
<td>L1</td>
<td>1000</td>
<td>02-OCT-95</td>
</tr>
<tr>
<td>2</td>
<td>Site1</td>
<td>Straight Time</td>
<td>L1</td>
<td>1000</td>
<td>25-SEP-95</td>
</tr>
<tr>
<td>3</td>
<td>Site1</td>
<td>Expense Reports</td>
<td>L1</td>
<td>1000</td>
<td>25-SEP-95</td>
</tr>
<tr>
<td>4</td>
<td>Site1</td>
<td>Expense Reports</td>
<td>L1</td>
<td>1001</td>
<td>09-OCT-95</td>
</tr>
<tr>
<td>5</td>
<td>Site1</td>
<td>Straight Time</td>
<td>L2</td>
<td>1001</td>
<td>09-OCT-95</td>
</tr>
</tbody>
</table>
If you submit Transaction Import for the transaction source *Site1* and do not specify a specific batch to process (pick all transactions with transaction source *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 L1 is shown in the following table:

<table>
<thead>
<tr>
<th>Transaction Number</th>
<th>Expenditure Type Class</th>
<th>Employee Number</th>
<th>Expenditure Ending Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Straight Time</td>
<td>1000</td>
<td>02-OCT-95</td>
</tr>
<tr>
<td>2</td>
<td>Straight Time</td>
<td>1000</td>
<td>25-SEP-95</td>
</tr>
<tr>
<td>3</td>
<td>Expense Reports</td>
<td>1000</td>
<td>25-SEP-95</td>
</tr>
<tr>
<td>4</td>
<td>Expense Reports</td>
<td>1001</td>
<td>09-OCT-95</td>
</tr>
</tbody>
</table>

Batch L2 is shown in the following table:

<table>
<thead>
<tr>
<th>Transaction Number</th>
<th>Expenditure Type Class</th>
<th>Employee Number</th>
<th>Expenditure Ending Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Straight Time</td>
<td>1001</td>
<td>09-OCT-95</td>
</tr>
<tr>
<td>6</td>
<td>Straight Time</td>
<td>1001</td>
<td>09-OCT-95</td>
</tr>
</tbody>
</table>

Because the transaction source has expenditure type classes *Straight Time* and *Expense Reports*, Transaction Import groups the transactions by employee, expenditure ending date, and expenditure type class when creating expenditures.

The resulting expenditures for batch L1 are shown in the following table:
The resulting expenditures for batch L2 are shown in the following table:

<table>
<thead>
<tr>
<th>Transaction Number</th>
<th>Expenditure Type</th>
<th>Employee Number</th>
<th>Expenditure Ending Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,6</td>
<td>Straight Time</td>
<td>1001</td>
<td>09-OCT-95</td>
</tr>
</tbody>
</table>

**Note:** Although 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.

Because 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:

<table>
<thead>
<tr>
<th>Batch Name</th>
<th>Expenditure Ending Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>09-OCT-95</td>
</tr>
<tr>
<td>L2</td>
<td>09-OCT-95</td>
</tr>
</tbody>
</table>

**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 
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.

In the example shown in the following table, all transactions have the transaction source 
Usage and expenditure type class Usages. The batch name is U1 and the expenditure 
ending date is 02-OCT-95.

**Note:** You do not need to enter an employee number for usage 
transactions.

<table>
<thead>
<tr>
<th>Transaction Number</th>
<th>Employee Number</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000</td>
<td>West</td>
</tr>
<tr>
<td>2</td>
<td>1000</td>
<td>East</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>West</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Midwest</td>
</tr>
</tbody>
</table>

Because all of these transactions have the same batch name, Oracle Projects creates only 
one expenditure batch, U1. For usage items, Transaction Import groups transactions by 
employee, organization, and expenditure ending date when creating expenditures. The 
resulting expenditures after import are shown in the following table:

<table>
<thead>
<tr>
<th>Trx Number</th>
<th>Expenditure Type Class</th>
<th>Employee Number</th>
<th>Organization</th>
<th>Expenditure Ending Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 2</td>
<td>Usages</td>
<td>1000</td>
<td><em>Employee Org</em></td>
<td>02-OCT-95</td>
</tr>
<tr>
<td>3</td>
<td>Usages</td>
<td></td>
<td>West</td>
<td>02-OCT-95</td>
</tr>
<tr>
<td>4</td>
<td>Usages</td>
<td></td>
<td>Midwest</td>
<td>02-OCT-95</td>
</tr>
</tbody>
</table>

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 and Processing Imported Transactions**

You can view and process imported transactions in various ways.

**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.

Tip: 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 IdentifyingTransactions

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.

Related Topics

Transaction Import Validation, page 13-25
The Transaction Import Interface Table, page 13-27
Populating the Interface Table, page 13-4
Importing Transactions, page 13-6
Resolving Import Exceptions, page 13-53

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.

Note: The transaction import validation logic is different when you run the process PRC: Interface Supplier Costs to import transactions from Oracle Purchasing and Oracle Payables. The processes uses predefined supplier cost transaction sources to import expenditure items and it rejects only the expenditure items that fail validation. The process
imports the valid expenditure items in the expenditure. You can use the
Review Transactions window to change the date for a rejected
expenditure item. Oracle Projects picks up the revised date for the
rejected transaction the next time that you run the process PRC:
Interface Supplier Costs.

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
13-53.

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 13-27.

- 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 13-27.

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.

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.

**Important:** 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
This section describes in detail each PA_TRANSACTION_INTERFACE_ALL column including validation and destination information.

**TRANSACTION_SOURCE**
A transaction source identifies the external source of the cost transaction.

**Validation:** Either a TRANSACTION_SOURCE or USER_TRANSACTION_SOURCE is required on all transactions. If a value is provided, then it must exist in PA_TRANSACTION_SOURCES.TRANSACTION_SOURCE.

**Destination:** PA_EXPENDITURE_GROUPS_ALL.TRANSACTION_SOURCE and PA_EXPENDITURE_ITEMS_ALL.TRANSACTION_SOURCE

**BATCH_NAME**
A batch name groups one or more expenditures into a single group.

**Validation:** This column is required on all transactions. All transactions in a batch must have the same transaction source.

**Destination:** Used to derive
PA_EXPENDITURE_GROUPS_ALL.EXPENDITURE_GROUP. Oracle Projects creates an expenditure group by concatenating BATCH_NAME, SYSTEM_LINKAGE_FUNCTION, and INTERFACE_ID.

**EXPENDITURE_ENDING_DATE**
An expenditure ending date is the last day of the expenditure week.

**Validation:** This column is required on all transactions. The expenditure ending date must be valid based on the expenditure cycle start day defined in Implementation Options. All transactions within an expenditure must have an expenditure item date that is on or before the expenditure ending date. All timecard transactions must have an expenditure item date within the expenditure week date range.
**Destination:** PA_EXPENDITURES_ALL.EXPENDITURE_ENDING_DATE

The maximum expenditure ending date of all expenditure items processed in a batch becomes the expenditure batch ending date.

**EMPLOYEE_NUMBER**

An *employee number* is the identifier of the employee that incurred the cost.

**Validation:** Either EMPLOYEE_NUMBER or PERSON_ID is required on labor and expense report transactions. These values are optional for transactions with other expenditure type classes. If you provide an EMPLOYEE_NUMBER, then it must exist in PER_ALL_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:** Used to derive PA_EXPENDITURES_ALL.INCURRED_BY_PERSON_ID

**ORGANIZATION_NAME**

An *organization name* is the name of the organization that incurred the expenditure.

**Validation:** Either the ORGANIZATION_NAME or ORGANIZATION_ID is required on usage transactions and is optional on other transactions. If you do not provide an ORGANIZATION_NAME, and you do provide an EMPLOYEE_NUMBER, then Transaction Import derives this value from the employee organization. If you provide a value, then it must exist in HR_ALL_ORGANIZATION_UNITS.NAME.

**Destination:** Used to derive PA_EXPENDITURES_ALL.INCURRED_BY_ORGANIZATION_ID. If you provide both an employee and an organization, then Transaction Import uses the employee information to derive the organization when the Import Employee Organization option is not used. The last employee assignment in the expenditure period is used to derive the employee organization.

**EXPENDITURE_ITEM_DATE**

An *expenditure item date* is the date cost is incurred.

**Validation:** This column is required on all transactions. The expenditure item date must be on or before the expenditure ending date. The expenditure item date on timecard transactions must fall within the expenditure week as defined by the expenditure ending date. For additional information about expenditure item date validations, see: Expenditure Item Validations, Oracle Project Costing User Guide.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.EXPENDITURE_ITEM_DATE

**PROJECT_NUMBER**

A *project number* is a unique identification number of the project that incurred the cost.

**Validation:** This column is required on all transactions and must exist in
PA_PROJECTS_ALL.SEGMENT1 and PA_PROJECTS_EXPEND_V. The project must have a project status that allows new transactions. The project must not be a project template. If multiple organization support is enabled, then the project must allow charges from the corresponding operating unit.

**Destination:** Used to derive PA_EXPENDITURE_ITEMS_ALL.PROJECT_ID and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_ID

### TASK_NUMBER

A *task number* is a unique identification number of the task within a project that incurred the cost.

**Validation:** This column is required on all transactions. The value must exist in PA_TASKS.TASK_NUMBER for the corresponding project number on the transaction and it must be a lowest level task that allows charges.

**Destination:** Used to derive PA_EXPENDITURE_ITEMS_ALL.TASK_ID and PA_COST_DISTRIBUTION_LINES_ALL.TASK_ID

### EXPENDITURE_TYPE

An *expenditure type* is an implementation-defined classification of the incurred cost.

**Validation:** This column is required on all transactions and must exist 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.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.EXPENDITURE_TYPE

### NON_LABOR_RESOURCE

A *non-labor resource* is an implementation-defined asset or pool of assets that incurred the cost.

**Validation:** This column is required for usage transactions. The value must exist 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

A *non-labor resource org name* is the name of the organization that owns the non-labor resource that incurred the cost.

**Validation:** Either the NON_LABOR_RESOURCE_ORG_NAME or NON_LABORRESOURCE_ORG_ID is required for usage transactions. If you provide the NON_LABORRESOURCE_ORG_NAME, then it is used to derive HR_ALL_ORGANIZATION_UNITS.ORGANIZATION_ID. The value must exist in PA_NON_LABORRESOURCE_OGNS.ORGANIZATION_ID for the specified
non-labor resource.

**Destination:** Used to derive PA_EXPENDITURE_ITEMS_ALL. ORGANIZATION_ID.

### QUANTITY

The *quantity* is the number of units for the transaction based on the unit of measure defined for the expenditure type.

**Validation:** This column is required on all transactions. QUANTITY, DENOM_RAW_COST, AND ACCT_RAW_COST must be zero for transactions with an expenditure type class of Burden Transaction.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.QUANTITY

### RAW_COST

*Raw cost* is the cost of the transaction in the project functional currency.

**Validation:** When you create transactions using the Review Transactions window, this column is automatically derived. However, Transaction Import does not require or use the value.

**Destination:** PA_EXPENDITURE_ITEMS_ALL_RAW_COST and PA_COST_DISTRIBUTION_LINES_ALL_AMOUNT

### EXPENDITURE_COMMENT

An *expenditure comment* describes the transaction in detail.

**Validation:** None

**Destination:** PA_EXPENDITURE_COMMENTS.EXPENDITURE_COMMENT

### TRANSACTION_STATUS_CODE

A *transaction status code* is a code that indicates the processing status of the transaction.

**Validation:** This column is required on all transactions and must be set to P for transactions to be imported.

**Destination:** None

### TRANSACTION_REJECTION_CODE

A *transaction rejection code* is a code that indicates a transaction was rejected by the Transaction Import program.

**Validation:** This column is system assigned.

**Destination:** None
EXPENDITURE_ID

An expenditure identifier is a unique system-assigned value that identifies expenditures created by Oracle Projects.

**Validation:** This column is system assigned and for internal use only.

**Destination:** PA_EXPENDITURES_ALL.EXPENDITURE_ID

ORIG_TRANSACTION_REFERENCE

An original transaction reference is a reference to the original transaction in the external system.

**Validation:** This column is required on all transactions. Unless the transaction source allows duplicate references, this reference, in combination with the transaction source, uniquely identifies the original transaction. An expenditure item must not already exist with the same identifier.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ORIG_TRANSACTION_REFERENCE

ATTRIBUTE_CATEGORY

An attribute category is the descriptive flexfield category for the descriptive flexfield information defined on the transaction.

**Validation:** Validated using the standard Oracle Applications Technology application programming interface (API) for validating attribute categories.

**Destination:** EXPENDITURE_ITEMS_ALL.ATTRIBUTE_CATEGORY

ATTRIBUTE1 through ATTRIBUTE10

An attribute is the descriptive flexfield information defined on the transaction.

**Validation:** The structure of the information that you enter in these columns (datatypes and value sets) should match the structure of the descriptive flexfield segments that you have defined for the transaction. Otherwise, you will experience validation problems when you try to access the information in the expenditure entry forms. These values are validated using the standard Oracle Applications Technology application programming interface (API) for validating descriptive flexfields. The following information applies to the validation of descriptive flexfield information during transaction import:

- You must populate this field with the attribute ID (code) rather than the meaning. The meaning will not pass the validation.

- The Transaction Import process validates descriptive flexfield attributes only if the Attribute Category field is populated.

- When you use the transaction import process to import a date as one of the segments in the Expenditure Items descriptive flexfield, the date must be in the
format RRRR/MM/DD HH24:MI:SS. If you attempt to import a date using any other format, then the transaction import process fails validation with the error PA_DFF_VALIDATION FAILED.

**Destination:** ATTRIBUTE1 through ATTRIBUTE10 in PA_EXPENDITURE_ITEMS_ALL

---

**RAW_COST_RATE**

A raw cost rate is the cost rate (raw cost divided by quantity) of the transaction in project functional currency.

**Validation:** This column is optional and is used only when the transaction source associated with the transaction is defined with the Import Raw Cost Amounts option enabled.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.RAW_COST_RATE

Oracle Projects uses this information for reporting purposes only.

---

**INTERFACE_ID**

An interface identifier is a unique system-assigned value that identifies transactions processed by a given concurrent request.

**Validation:** This column is system assigned and is for internal use only.

**Destination:** None

---

**UNMATCHED_NEGATIVE_TXN_FLAG**

An unmatched negative transaction flag is an attribute that indicates that the transaction has a negative amount and should not be matched to an expenditure item that already exists in the system.

**Validation:** Possible values are Y, N, or null. If this column is set to Y, then Transaction Import bypasses the matching validation logic that is usually executed for adjustments (negative transactions).

**Destination:** If this column is set to N or null, then Transaction Import finds the matching expenditure item and populates PA_EXPENDITURE_ITEMS_ALL.ADJUSTED_EXPENDITURE_ITEM_ID with the EXPENDITURE_ITEM_ID from the matching expenditure item.

---

**EXPENDITURE_ITEM_ID**

An expenditure item identifier is a unique system-assigned value that identifies expenditure items created by Oracle Projects.

**Validation:** This column is system assigned and for internal use only.

**Destination:** PA_EXPENDITURES_ALL.EXPENDITURE_ITEM_ID
**ORG_ID**

An organization identifier is a unique system-assigned value that identifies an organization classified as an operating unit. If a user enters transactions in the Review Transactions window, then this column is populated with the ORGANIZATION_ID of the operating unit defined for the user.

**Validation:** If multi-organization support is implemented, then a value is required. The value it must exist in HR_ALL_ORGANIZATION_UNITS.ORGANIZATION_ID and HR_ORGANIZATION_INFORMATION.ORGANIZATION_ID and have an ORG_INFORMATION_CONTEXT of Operating Unit Information.

**Destination:** PA_EXPENDITURE_GROUPS_ALL.ORG_ID, PA_EXPENDITURES_ALL.ORG_ID, PA_EXPENDITURE_ITEMS_ALL.ORG_ID, and PA_COST_DISTRIBUTION_LINES_ALL.ORG_ID.

**DR_CODE_COMBINATION_ID**

A debit code combination identifier is a unique system-assigned value that identifies a General Ledger account used to record the debit side of an accounting entry. If the transaction source associated with the transaction has the Allow Adjustments option enabled, then Oracle Projects uses this value to create reversing accounting entries.

**Validation:** If the transaction is associated with a transaction source that has the Raw Cost GL Accounted option enabled, then a value is required and it must exist in GL_CODE_COMBINATIONS.CODE_COMBINATION_ID.

**Destination:** PA_COST_DISTRIBUTION_LINES_ALL.DR_CODE_COMBINATION_ID

**CR_CODE_COMBINATION_ID**

A credit code combination identifier is a unique system-assigned value that identifies a General Ledger account used to record the credit side of an accounting entry. If the transaction source associated with the transaction has the Allow Adjustments option enabled, and is not a supplier cost transaction source, then Oracle Projects uses this value to create reversing accounting entries. For supplier cost transaction sources, this value is used only for informational purposes in Oracle Projects.

**Validation:** If the transaction is associated with a transaction source that has the Raw Cost GL Accounted option enabled, then a value is required and must exist in GL_CODE_COMBINATIONS.CODE_COMBINATION_ID.

**Destination:** PA_COST_DISTRIBUTION_LINES_ALL.CR_CODE_COMBINATION_ID

**CDL_SYSTEM_REFERENCE1**

A cost distribution line system reference is a reference to a record in an external system.

**Validation:** For supplier invoice transactions created in Oracle Payables, the column holds PO_VENDORS.VENDOR_ID. For payment and discount transactions created in Oracle Payables, the column holds
AP_PAYMENT_HIST_DISTS.PAYMENT_HIST_DIST_ID. For transactions associated with other transaction sources that have the Raw Cost GL Accounted option enabled, a value is optional. For transactions associated with transaction sources that do not have the Raw Cost GL Accounted option enabled, if you provide a value, then it is ignored.

**Destination:** PA_COST_DISTRIBUTION_LINES_ALL.SYSTEM_REFERENCE1, except when the transaction represents an Oracle Payables payment or discount, in which case the destination is PA_COST_DISTRIBUTION_LINES_ALL.SYSTEM_REFERENCE5.

**CDL_SYSTEM_REFERENCE2**

A cost distribution line system reference is a reference to a record in an external system. For transactions that represent supplier invoices in Oracle Payables, the reference is used to associate the transaction with an invoice number in Oracle Payables. For transactions that represent receipt accruals in Oracle Purchasing, the reference is used to associate the transaction with a purchase order.

**Validation:** For transactions created by Oracle Payables, the column holds AP_INVOICES_ALL.INVOICE_ID. For transactions created by Oracle Purchasing, the column holds PO_HEADERS_ALL.PO_HEADER_ID. For transactions associated with other transaction sources that have the Raw Cost GL Accounted option enabled, a value is optional. For transactions associated with transaction sources that do not have the Raw Cost GL Accounted option enabled if you provide a value, then it is ignored.

**Destination:** PA_COST_DISTRIBUTION_LINES_ALL.SYSTEM_REFERENCE and, for transactions created by Oracle Payables or Oracle Purchasing, it is also copied to PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_HEADER_ID.

**CDL_SYSTEM_REFERENCE3**

A cost distribution line system reference is a reference to a record in an external system. For transactions that originate in Oracle Payables, the reference is used to associate the transaction with an invoice line number in Oracle Payables. For transactions that originate in Oracle Purchasing, the reference is used to associate the transaction with a purchase order distribution.

**Validation:** For transactions created by Oracle Payables, the column holds AP_INVOICE_LINES_ALL.INVOICE_LINE_NUMBER. For transactions created by Oracle Purchasing, the column holds PO_HEADERS_ALL.PO_DISTRIBUTION_ID. For transactions associated with other transaction sources that have the Raw Cost GL Accounted option enabled, a value is optional. For transactions associated with transaction sources that do not have the Raw Cost GL Accounted option enabled if you provide a value, then it is ignored.

**Destination:** PA_COST_DISTRIBUTION_LINES_ALL.SYSTEM_REFERENCE3, and PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_LINE_NUMBER

**GL_DATE**

A general ledger date is date the transaction is accounted in General Ledger.
**Validation:** If the transaction is associated with a transaction source that has the Raw Cost GL Accounted option enabled, then a GL date is required.

**Destination:** PA_COST_DISTRIBUTION_LINES_ALL.GL_DATE
Oracle Projects uses this information for reporting purposes only.

**BURDENED_COST**

A *burdened cost* is a cost amount that represents the sum of the raw cost plus a burden cost in the project functional currency code.

**Validation:** When you create transactions using the Review Transactions window, this column is automatically derived. However, Transaction Import does not use or require this value.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.BURDENED_COST

**BURDENED_COST_RATE**

A *burdened cost rate* is the burden cost rate (burdened cost divided by quantity) of the transaction in project functional currency.

**Validation:** This column is optional when transactions have an expenditure type class of Burden Transaction, or when transactions are associated with a transaction source that has the Import Burdened Amounts option enabled. Burden transactions are required to have QUANTITY, DENOM_RAW_COST, and ACCT_RAW_COST attributes equal to zero. For all other transactions, the value is ignored.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.BURDEN_COST_RATE
Oracle Projects uses this information for reporting purposes only.

**SYSTEM_LINKAGE**

A *system linkage* is a system-defined value that indicates the expenditure type class of the transaction.

**Validation:** This column is optional. If you do not provide a value, then the default system linkage defined for the transaction source is used. If you provide a value, then the value must exist in PA_EXPENDITURE_TYPES.SYSTEM_LINKAGE_FUNCTION and PA_EXPEND_TYP_SYS_LINKS.SYSTEM_LINKAGE_FUNCTION for the corresponding EXPENDITURE_TYPE on the transaction.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.SYSTEM_LINKAGE_FUNCTION

**TXN_INTERFACE_ID**

A *transaction interface identifier* is a unique system-defined value that identifies each transaction loaded into the interface table.

**Validation:** This column is system assigned and for internal use only.

**Destination:** None
**USER_TRANSACTION_SOURCE**

A *user transaction source* is a user-defined name for a transaction source.

**Validation:** This column is required when a TRANSACTION_SOURCE value is not provided. If a value is provided, then it must exist in PA_TRANSACTION_SOURCES.USER_TRANSACTION_SOURCE.

**Destination:** This value is used to derive PA_EXPENDITURE_GROUPS_ALL.TRANSACTION_SOURCE and PA_EXPENDITURE_ITEMS_ALL.TRANSACTION_SOURCE.

**CREATED_BY**

The *created by* attribute represents the system-assigned identifier of the user that created transaction.

**Validation:** This column is required and must be a number.

**Destination:** When transactions are created using the Review Transactions window, the FND_USERS.USER_ID defined for the user creating the transactions is stored in this column. However, this value is not used when Transaction Import creates expenditure items for the transaction. Transaction Import uses the FND_USERS.USER_ID defined for the user that runs the Transaction Import concurrent program to populate the CREATED_BY attribute in the PA_EXPENDITURE_GROUPS_ALL, PA_EXPENDITURES_ALL, and PA_EXPENDITURE_ITEMS_ALL tables.

**CREATION_DATE**

A *creation date* is the date the transaction is created in the system.

**Validation:** This column is required and must be a valid date.

**Destination:** When you create transactions using the Review Transactions window, the system date is stored in this column. However, this value is not used when Transaction Import creates expenditure items for the transaction. When you run the Transaction Import concurrent program, the program uses the system date to populate the CREATION_DATE attribute in the PA_EXPENDITURE_GROUPS_ALL, PA_EXPENDITURES_ALL, and PA_EXPENDITURE_ITEMS_ALL tables.

**LAST_UPDATED_BY**

The *last updated by* attribute represents the system-defined identifier of the user that last updated the transaction.

**Validation:** This column is required and must be a number.

**Destination:** When transactions are updated using the Review Transactions window, the FND_USERS.USER_ID defined for the user updating the transactions is stored in this column. However, this value is not used when Transaction Import creates expenditure items for the transaction. Transaction Import uses the
FND_USERS.USER_ID defined for the user that runs the Transaction Import concurrent program to populate the LAST_UPDATED_BY attribute in the PA_EXPENDITURE_GROUPS_ALL, PA_EXPENDITURES_ALL, and PA_EXPENDITURE_ITEMS_ALL tables.

**LAST_UPDATE_DATE**

A *last update date* is the date the transaction is last updated in the system.

**Validation:** This column is required and must be a valid date.

**Destination:** When you update transactions using the Review Transactions window, the system date is stored in this column. However, this value is not used when Transaction Import creates expenditure items for the transaction. When you run Transaction Import concurrent program, the program uses the system date to populate the LAST_UPDATE_DATE attribute in the PA_EXPENDITURE_GROUPS_ALL, PA_EXPENDITURES_ALL, and PA_EXPENDITURE_ITEMS_ALL tables.

**RECEIPT_CURRENCY_AMOUNT**

A *receipt currency amount* is the amount of an expense report transaction in the original currency.

**Validation:** This column is required when the SYSTEM_LINKAGE_FUNCTION is Expense Reports and the RECEIPT_CURRENCY_CODE is different from the DENOM_CURRENCY_CODE.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.RECEIPT_CURRENCY_AMOUNT

**RECEIPT_CURRENCY_CODE**

A *receipt currency code* is the currency code for the expense report receipt currency.

**Validation:** This column is optional and only used when the SYSTEM_LINKAGE_FUNCTION is Expense Reports. If you provide a value, then it must exist in FND_CURRENCIES_VL.CURRENCY_CODE. In addition, the FND_CURRENCIES_VL.ENABLED_FLAG must be set to Y and active as of the expenditure item date on the transaction.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.RECEIPT_CURRENCY_CODE

**RECEIPT_EXCHANGE_RATE**

A *receipt exchange rate* is the exchange rate used to convert the receipt currency to the transaction (reimbursement) currency for expense reports.

**Validation:** This column is optional and only used when the SYSTEM_LINKAGE_FUNCTION is Expense Reports.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.RECEIPT_EXCHANGE_RATE
DENOM_CURRENCY_CODE

A denominated currency code is the code of the currency used for the transaction.

**Validation:** This column is required on all transactions and must be a valid FND_CURRENCIES_VL.CURRENCY_CODE. In addition, the FND_CURRENCIES_VL.ENABLED_FLAG must be set to Y as of the expenditure item date.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.DENOM_CURRENCY_CODE and PA_COST_DISTRIBUTION_LINES_ALL.DENOM_CURRENCY_CODE

DENOM_RAW_COST

A denominated raw cost is the raw cost amount in the transaction currency.

**Validation:** This column is required when the transaction source is defined with the Import Raw Cost Amounts option enabled. QUANTITY, DENOM_RAW_COST, AND ACCT_RAW_COST attributes must be zero for transactions with an expenditure type class of Burden Transaction.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.DENOM_RAW_COST and PA_COST_DISTRIBUTION_LINES_ALL.DENOM_RAW_COST

DENOM_BURDENED_COST

A denominated burdened cost is the burdened cost amount in the transaction currency.

**Validation:** This column is required for transactions with an expenditure type class of Burden Transaction, and for transactions that are associated with a transaction source that has the Import Burdened Amounts option enabled. Burden transactions are required to have QUANTITY, DENOM_RAW_COST, ACCT_RAW_COST attributes equal to zero. For all other transactions, the value is ignored.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.DENOM_BURDENED_COST and PA_COST_DISTRIBUTION_LINES_ALL.DENOM_BURDENED_COST

ACCT_RATE_DATE

An accounted rate date is the date used for converting the transaction to the functional currency of the operating unit.

**Validation:** This column is required when the transaction source associated with the transaction has the Raw Cost GL Accounted option enabled. This column is optional when the ACCT_RATE_TYPE on the transaction is User.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ACCT_RATE_DATE and PA_COST_DISTRIBUTION_LINES_ALL.ACCT_RATE_DATE

ACCT_RATE_TYPE

An accounted rate type is the conversion type used to convert the transaction to the
functional currency of the operating unit.

**Validation:** This column is optional when the transaction source associated with the transaction has the Raw Cost GL Accounted option enabled. The value must exist in PA_CONVERSION_TYPES.V.CONVERSION_TYPE.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ACCT_RATE_TYPE and PA_COST_DISTRIBUTION_LINES_ALL.ACCT_RATE_TYPE

### ACCT.Exchange Rate

An **accounted exchange rate** is the rate used to convert the transaction to the functional currency of the operating unit.

**Validation:** This column is optional when the transaction source associated with the transaction has the Raw Cost GL Accounted option enabled and the ACCT_RATE_TYPE on the transaction is *User*.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ACCT.Exchange RATE and PA_COST_DISTRIBUTION_LINES_ALL.ACCT.Exchange RATE

### ACCT.Raw Cost

An **accounted raw cost** is the raw cost converted to the functional currency of the operating unit.

**Validation:** This column is required when the transaction source associated with the transaction has the Raw Cost GL Accounted option enabled. The functional raw cost is calculated using the provided ACCT_RATE_DATE and ACCT_RATE_TYPE. The calculated value must be within the ACCT.Exchange_ROUNDING_LIMIT of the provided ACCT.Raw Cost. For all other transactions, the value is ignored.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ACCT.Raw Cost and PA_COST_DISTRIBUTION_LINES_ALL.ACCT.Raw Cost

### ACCT.Burdened Cost

An **accounted burdened cost** is the burdened cost converted to the functional currency of the operating unit.

**Validation:** This column is required when:

- The transaction source associated with the transaction has the Raw Cost GL Accounted option enabled.

- Transactions either have an expenditure type class of Burden Transaction, or are associated with a transaction source that has the Import Burdened Amounts option enabled. Burden transactions are required to have QUANTITY, DENOM.Raw Cost, and ACCT.Raw Cost attributes equal to zero.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ACCT.Burdened Cost
ACCT_EXCHANGE_ROUNDING_LIMIT

An **accounted exchange rounding limit** is the rounding limit of the functional currency of the operating unit.

**Validation:** This column is optional. If the derivation of the functional currency raw cost is within the rounding limit, then the transaction is accepted, otherwise it is rejected. If the value of ACCT_EXCHANGE_ROUNDING_LIMIT is null, then the rounding limit value used is zero.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ACCT_ROUNDING_LIMIT

PROJECT_CURRENCY_CODE

A **project currency code** is the code of the currency defined for the project.

**Validation:** This column is optional. If you do not provide a value, then Transaction Import derives the value.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJECT_CURRENCY_CODE and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_CURRENCY_CODE

PROJECT_RATE_DATE

A **project rate date** is the date used to convert the transaction to the project currency.

**Validation:** This column is optional.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJECT_RATE_DATE and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_RATE_DATE

PROJECT_RATE_TYPE

A **project rate type** is the rate type used for converting the transaction to the project currency.

**Validation:** This column is optional. If you provide a value, then it must exist in PA_CONVERSION_TYPES_V.CONVERSION_TYPE.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJECT_RATE_TYPE and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_RATE_TYPE

PROJECT_EXCHANGE_RATE

A **project exchange rate** is the rate used to convert the transaction to project currency.

**Validation:** This column is required when the PROJECT_RATE_TYPE is set to User.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJECT_EXCHANGE_RATE and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_EXCHANGE_RATE
**ORIG_EXP_TXN_REFERENCE1**

An original expenditure transaction reference is an identifier of a transaction in an external system.

**Validation:** For transactions created in Oracle Payables, this column contains the value of AP_INVOICE_DISTRIBUTIONS_ALL.INVOICE_ID. For all other transactions, this column is not validated and is optional.

**Destination:** PA_EXPENDITURES_ALL.ORIG_EXP_TXN_REFERENCE1

This column is used to determine how to group transactions into an expenditure group. The following attributes are used in the following order to determine how to group transactions into an expenditure group:

1. EXPENDITURE_ENDING_DATE
2. INCURRED_BY_PERSON_ID
3. ORGANIZATION_ID
4. ORIG_EXP_TXN_REFERENCE1
5. ORIG_USER_EXP_TXN_REFERENCE
6. VENDOR_ID
7. ORIG_EXP_TXN_REFERENCE2
8. ORIG_EXP_TXN_REFERENCE3
9. ACCRUAL_FLAG
10. PERSON_TYPE

**ORIG_EXP_TXN_REFERENCE2**

An original expenditure transaction reference is an identifier of a transaction in an external system.

**Validation:** This column is not validated and is optional.

**Destination:** PA_EXPENDITURES_ALL.ORIG_EXP_TXN_REFERENCE2

This column is used to determine how to group transactions into an expenditure group. For additional information about how expenditure groups are defined, see: ORIG_EXP_TXN_REFERENCE1, page 13-42.

**ORIG_EXP_TXN_REFERENCE3**

An original expenditure transaction reference is an identifier of a transaction in an external system.
system.

**Validation:** This column is not validated and is optional.

**Destination:** PA_EXPENDITURES_ALL.ORIG_EXP_TXN_REFERENCE3

This column is used to determine how to group transactions into an expenditure group. For additional information about how expenditure groups are defined, see: ORIG_EXP_TXN_REFERENCE1, page 13-42.

### ORIG_USER_EXP_TXN_REFERENCE

An original user expenditure transaction reference is an identifier of a transaction in an external system.

**Validation:** For transactions created in Oracle Payables, this column contains the value of AP_INVOICES_ALL.INVOICE_NUM. For all other transactions, this column is not validated and is optional.

**Destination:** PA_EXPENDITURES_ALL.ORIG_USER_EXP_TXN_REFERENCE

This column is used to determine how to group transactions into an expenditure group. For additional information about how expenditure groups are defined, see: ORIG_EXP_TXN_REFERENCE1, page 13-42.

### VENDOR_NUMBER

A vendor number is a unique identification number of the supplier that provided the goods or services for the transaction.

**Validation:** Either VENDOR_NUMBER or VENDOR_ID is required on transactions that have expenditure type class of Supplier Invoice. If you provide a VENDOR_NUMBER, then it must exist in PO_VENDORS.SEGMENT1.

**Destination:** This column is used to derive PA_EXPENDITURES_ALL.VENDOR_ID.

### OVERRIDE_TO_ORGANIZATION_NAME

An override to organization name is the name of the organization that incurred the cost.

**Validation:** This column is optional and you can provide either an OVERRIDE_TO_ORGANIZATION_NAME or OVERRIDE_TO_ORGANIZATION_ID. If you provide an OVERRIDE_TO_ORGANIZATION_NAME, then it must exist in HR_ALL_ORGANIZATION_UNITS.NAME.

**Destination:** This column is used to derive PA_EXPENDITURE_ITEMS_ALL.OVERRIDE_TO_ORGANIZATION_ID.

### REVERSED_ORIG_TXN_REFERENCE

A reversed original transaction reference indicates that the transaction reverses a transaction that already exists in the system.

**Validation:** This column is optional. If you provide a value, then it must exist in
PA_EXPENDITURE_ITEMS_ALL.ORIG_TRANSACTION_REFERENCE.

**Destination:** This column is used to derive
PA_EXPENDITURE_ITEMS_ALL.ADJUSTED_EXPENDITURE_ITEM_ID.

**BILLABLE_FLAG**

A *billable flag* is an attribute that indicates whether the transaction is billable when the project is a contract project, or whether the transaction is capitalizable when the project is a capital project.

**Validation:** Possible values are Y, N, and null.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.BILLABLE_FLAG

If you do not provide a value, then the value is derived based on task and transaction control settings.

**PERSON_BUSINESS_GROUP_NAME**

A *person business group name* is the name of an organization that has a business group classification and is defined for the person that incurred the cost.

**Validation:** This column is required when an employee is defined in more than one business group. If you provide a value is provided, then it must exist in HR_ALL_ORGANIZATION_UNITS.NAME and in HR_ORGANIZATION_INFORMATION.ORGANIZATION_ID and have an ORG_INFORMATION_CONTEXT of Business Group Information. Alternatively, you can provide the value of PERSON_BUSINESS_GROUP_ID instead of PERSON_BUSINESS_GROUP_NAME.

**Destination:** If you provide a value, then it is used to derive a PERSON_BUSINESS_GROUP_ID which, in turn, is used to derive a PA_EXPENDITURES_ALL.INCURRED_BY_PERSON_ID.

**PROJFUNC_CURRENCY_CODE**

A *project functional currency code* is the code of the functional currency defined for the project.

**Validation:** This column is optional. If you do not provide a value, then Transaction Import derives the value.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJFUNC_CURRENCY_CODE

**PROJFUNC_COST_RATE_TYPE**

A *project functional cost rate type* is the rate type used for converting the transaction to project functional currency.

**Validation:** This column is optional. If you provide a value, then it must exist in PA_CONVERSION_TYPES_V.CONVERSION_TYPE.
**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJFUNC_COST_RATE_TYPE and PA_COST_DISTRIBUTION_LINES_ALL.PROJFUNC_COST_RATE_TYPE

**PROJFUNC_COST_RATE_DATE**

*Project functional cost rate date* is the date used to convert the transaction to the project functional currency.

**Validation:** This column is optional.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJFUNC_COST_RATE_DATE and PA_COST_DISTRIBUTION_LINES_ALL.PROJFUNC_COST_RATE_DATE

**PROJFUNC_COST_EXCHANGE_RATE**

*Project functional cost exchange rate* is the rate used to convert the transaction to project functional currency.

**Validation:** This column is required when the PROJFUNC_COST_RATE_TYPE is set to *User*.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJFUNC_COST_EXCHANGE_RATE and PA_COST_DISTRIBUTION_LINES_ALL.PROJFUNC_COST_EXCHANGE_RATE

**PROJECT_RAW_COST**

*Project raw cost* is the cost of the transaction in the project currency.

**Validation:** When you create transactions using the Review Transactions window, this column is automatically derived. However, Transaction Import does not use or require a value for this column.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJECT_RAW_COST and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_RAW_COST

**PROJECT_BURDENED_COST**

*Project burdened cost* is a cost amount that represents the sum of the raw cost plus burden cost in the project currency code.

**Validation:** When you create transactions using the Review Transactions window, this column is automatically derived. However, Transaction Import does not use or require a value for this column.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.BURDENED_COST

**ASSIGNMENT_NAME**

An assignment name is the name of an assignment. An assignment is a work position on a project that is associated with a specific person resource.

**Validation:** This column is optional and is used only when the transaction is a timecard or expense report. You can provide either the ASSIGNMENT_ID or
ASSIGNMENT_NAME. If you provide an ASSIGNMENT_NAME, then it used to derive the ASSIGNMENT_ID and it must exist in PA_PROJECT_ASSIGNMENTS.ASSIGNMENT_ID.

**Destination:** This column is used to derive PA_EXPENDITURE_ITEMS_ALL.ASSIGNMENT_ID.

**WORK_TYPE_NAME**

A work type name is the name of a work type. A work type is an implementation-defined classification of work.

**Validation:** This column is optional and you can provide either a WORK_TYPE_NAME or WORK_TYPE_ID. If you provide a WORK_TYPE_NAME, then it is used to derive a WORK_TYPE_ID and it must exist in PA_WORK_TYPES.WORK_TYPE_ID.

**Destination:** This column is used to derive PA_EXPENDITURE_ITEMS_ALL.WORK_TYPE_ID and PA_COST_DISTRIBUTION_LINES_ALL.WORK_TYPE_ID.

**CDL_SYSTEM_REFERENCE4**

A cost distribution line system reference is a reference to a record in an external system.

**Validation:** When the transaction is a payment or discount created by Oracle Payables, the column holds AP_INVOICE_PAYMENTS.INVOICE_PAYMENT_ID. When the transaction is created by Oracle Purchasing, the column holds RCV_TRANSACTIONS.TRANSACTION_ID. For transactions associated with other transaction sources that have the Raw Cost GL Accounted option enabled, a value is optional. For transactions associated with transaction sources that do not have the Raw Cost GL Accounted option enabled, if you provide a value, then it is ignored.

**Destination:** COST_DISTRIBUTION_LINES_ALL.SYSTEM_REFERENCE4

When the transaction is a payment or discount created by Oracle Payables, it is also copied to the PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_PAYMENT_ID column. When the transaction is created by Oracle Purchasing, it is also copied to PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_DISTRIBUTION_ID.

**ACCRUAL_FLAG**

An accrual flag is an attribute that indicates if a transaction is a period end accrual.

**Validation:** This column is optional. Possible values are Y, N, and null.

**Destination:** PA_EXPENDITURE_GROUPS_ALL.PERIOD_ACCRUAL_FLAG

**PROJECT_ID**

A project identifier is a unique system-assigned value that identifies the project number.
**Validation:** This column is system assigned and for internal use only.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PROJECT_ID and PA_COST_DISTRIBUTION_LINES_ALL.PROJECT_ID

**TASK_ID**

A task identifier is a unique system-assigned value that identifies the task number.

**Validation:** This column is system assigned and for internal use only.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.TASK_ID and PA_COST_DISTRIBUTION_LINES_ALL.TASK_ID

**PERSON_ID**

A person identifier is a unique system-assigned value that identifies the person that incurred the cost.

**Validation:** Either EMPLOYEE_NUMBER or PERSON_ID is required on labor and expense report transactions, but is optional for transactions with other expenditure type classes. If you provide the PERSON_ID, then it must exist in PER_ALL_PEOPLE_F.PERSON_ID. 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_ID**

An organization identifier is a unique system-assigned value that identifies the organization that incurred the expenditure.

**Validation:** Either the ORGANIZATION_NAME or ORGANIZATION_ID is required on usage transactions and is optional on other transactions. If you provide an EMPLOYEE_NUMBER, then this column can be null, in which case Transaction Import derives this value from the employee organization. If you provide an ORGANIZATION_ID, then it must exist in HR_ALL_ORGANIZATION_UNITS.ORGANIZATION_ID.

**Destination:** PA_EXPENDITURES_ALL.INCURRED_BY_ORGANIZATION_ID

The last employee assignment in the expenditure period is used to derive the organization.

**NON_LABOR_RESOURCE_ORG_ID**

A non-labor resource organization identifier is a unique system-assigned value that identifies the organization that owns a non-labor resource that incurred the cost.

**Validation:** Either the NON_LABOR_RESOURCE_ORG_NAME or NON_LABOR_RESOURCE_ORG_ID is required for usage transactions. If you provide
NON_LABORRESOURCE ORG ID, then it must exist in
PA_NON_LABORRESOURCE ORG S.ORGANIZATION ID for the specified
non-labor resource.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ORGANIZATION ID

**VENDOR ID**

A vendor identifier is a unique system-assigned value that identifies the supplier that
provided the goods or services associated with the transaction.

**Validation:** This column is required on transactions that have expenditure type class of
Supplier Invoice if you do not provide a VENDOR_NUMBER. If you provide a value,
then it must exist in PO_VENDORS.VENDOR ID.

**Destination:** PA_EXPENDITURES_ALL.VENDOR ID

**OVERRI D TO ORGANIZATION ID**

An override to organization identifier is a unique system-assigned value that identifies the
organization that incurred the cost.

**Validation:** For transactions created in Oracle Payables, this column holds
AP_INVOICE_DISTRIBUTIONS_ALL.EXPEN DI TURE_ORGANIZATION ID. For
other transactions this column is optional and you can provide either an
OVERRI D TO ORGANIZATION NAME or OVERRI D TO ORGANIZATION ID.
If you provide an OVERRI D TO ORGANIZATION ID, then it must exist in
HR_ALL_ORGANIZATION_UNITS.ORGANIZATION ID.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.OVERRIDE TO ORGANIZATION ID

**ASSIGNMENT ID**

An assignment identifier is a unique system-assigned value that identifies an assignment.
An assignment is a work position on a project that is associated with a specific person
resource.

**Validation:** This column is optional and is used only when the transaction is a timecard
or expense report. You can provide either the ASSIGNMENT_ID or
ASSIGNMENT NAME. If you provide an ASSIGNMENT_ID, then it must exist in
PA_PROJECT_ASSIGNMENTS.ASSIGNMENT ID.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ASSIGNMENT ID

**WORK TYPE ID**

A work type identifier is a unique system-assigned value that identifies the work type. A
work type is an implementation-defined classification of work.

**Validation:** This column is optional and you can provide either a
WORK_TYPE NAME or WORK_TYPE ID. If you provide a WORK_TYPE ID, then it
must exist in PA_WORK TYPES.WORK_TYPE ID.
**Destination**: PA_EXPENDITURE_ITEMS_ALL.WORK_TYPE_ID

**PERSON_BUSINESS_GROUP_ID**

A *person business group identifier* is a unique system-assigned value that identifies an organization that has a business group classification and is defined for the person that incurred the cost.

**Validation**: This column is required when an employee is defined in more than one business group. If you provide a value, then it must exist in HR_ALL_ORGANIZATION_UNITS.ORGANIZATION_ID and in HR_ORGANIZATION_INFORMATION.ORGANIZATION_ID and have an ORG_INFORMATION_CONTEXT of Business Group Information. Alternatively, you can provide the value of PERSON_BUSINESS_GROUP_NAME instead of PERSON_BUSINESS_GROUP_ID.

**Destination**: If you provide a value, then this column is used to derive PA_EXPENDITURES_ALL.INCURRED_BY_PERSON_ID.

**INVENTORY_ITEM_ID**

An *inventory item identifier* is a unique system-defined value that identifies inventory items.

**Validation**: This column is system assigned on transactions created by Oracle Project Manufacturing and is for internal use only.

**Destination**: PA_EXPENDITURE_ITEMS_ALL.INVENTORY_ITEM_ID

**WIP_RESOURCE_ID**

A *work-in-process resource identifier* is a unique system-defined value that identifies labor or non-labor resources on work-in-process transactions.

**Validation**: This column is system assigned on transactions created by Oracle Project Manufacturing and is for internal use only.

**Destination**: PA_EXPENDITURE_ITEMS_ALL.WIP_RESOURCE_ID

**UNIT_OF_MEASURE**

A *unit of measure* is a classification created in Oracle Project Manufacturing.

**Validation**: This column is system assigned and used only on transactions created in Oracle Project Manufacturing.

**Destination**: PA_EXPENDITURE_ITEMS_ALL.UNIT_OF_MEASURE

**PO_NUMBER**

A *purchase order number* identifies purchase order documents created in Oracle Purchasing.
**Validation:** This column is optional and you can provide either PO_NUMBER or PO_HEADER_ID. When you provide a value, it is only used when the transaction is for a contingent worker. If you provide values for both columns, then PO_HEADER_ID is used during validation. If you provide a PO_NUMBER, then it must exist in PO_HEADERS_ALL.SEGMENT1.

**Destination:** None

**PO_HEADER_ID**

A *purchase order identifier* is a system-assigned unique value that identifies purchase order documents created in Oracle Purchasing.

**Validation:** This column is optional and you can provide either the PO_NUMBER or PO_HEADER_ID. When you provide a value, it is only used when the transaction is for a contingent worker. If you provide values for both columns, then the PO_HEADER_ID is used during validation. If you provide a PO_HEADER_ID is provided, then it must exist in PO_HEADERS_ALL.HEADER_ID.

**Destination:** None

**PO_LINE_NUM**

A *purchase order line number* identifies specific lines on a purchase order document created by Oracle Purchasing.

**Validation:** This column is optional and you can provide either the PO_LINE_NUM or the PO_LINE_ID. When you provide a value, it is only used when the transaction is for a contingent worker. If you provide values for both columns, then the PO_LINE_ID is used during validation. If you provide the PO_LINE_NUM, then it must exist in PO_LINES_ALL.LINE_NUM for the purchase order specified by PO_NUMBER or PO_HEADER_ID. At least one distribution associated with the purchase order line must be related to the project and task specified for the transaction.

**Destination:** Used to derive to derive PA_EXPENDITURE_ITEMS_ALL.PO_LINE_ID.

**PO_LINE_ID**

A *purchase order line identifier* is a system-defined unique value that identifies individual lines on purchase order documents created in Oracle Purchasing.

**Validation:** This column is optional and you can provide either the PO_LINE_NUM or the PO_LINE_ID. When you provide a value, it is only used when the transaction is for a contingent worker. If you provide values for both columns, then the PO_LINE_ID is used during validation. If you provide the PO_LINE_ID, then it must exist in PO_LINES_ALL.PO_LINE_ID for the purchase order specified by PO_NUMBER or PO_HEADER_ID. At least one distribution associated with the purchase order line must be related to the project and task specified for the transaction.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PO_LINE_ID
PERSON_TYPE

A person type is the type of person that incurred the cost for the transaction.

**Validation:** This column is optional for timecard and expense reports. Possible values are:

- CWK - Contingent worker
- EMP - Employee
- null

If you do not provide a value, then the system sets the value to EMP.

**Destination:** None

PO_PRICE_TYPE

A purchase order price type is an attribute that identifies the contingent worker rate for the transaction.

**Validation:** This column is optional. If you provide a value, then it is only used when the transaction is for a contingent worker and it must exist in PO_LINES_ALL.PRICE_TYPE_LOOKUP_CODE for the purchase order line defined on the transaction.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.PO_PRICE_TYPE

ADJUSTED_EXPENDITURE_ITEM_ID

An adjusted expenditure item identifier is a unique system-assigned value that indicates that the transaction reverses another transaction and refers to the original transaction.

**Validation:** This column is system assigned and for internal use only.

**Destination:** PA_EXPENDITURE_ITEMS_ALL.ADJUSTED_EXPENDITURE_ITEM_ID

FC_DOCUMENT_TYPE

A funds check document type is a system-assigned attribute that indicates whether the transaction is required to undergo funds checking.

**Validation:** This column is system assigned and is for internal use only on supplier cost transactions created in Oracle Payables or Oracle Purchasing. The valid values are as follows:

- CMT - Commitments
- ACT - Actuals
- ALL - Commitments and Actuals
• NOT - None

Destination: None

**DOCUMENT_TYPE**

A *document type* is a system-assigned attribute that identifies the type of document created in Oracle Payables or Oracle Purchasing.

**Validation:** This column is system assigned and for internal use only. If the transaction was created by Oracle Payables, then the column holds AP_INVOICES_ALL.INVOICE_TYPE_LOOKUP_CODE. If the transaction was created by Oracle Purchasing, then the column holds RCV_TRANSACTIONS.DESTINATION_TYPE_CODE. If a value is provided on any other type of transaction, then it is ignored.

Destination: PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_TYPE

**DOCUMENT_DISTRIBUTION_TYPE**

A *document distribution type* is a system-assigned attribute that indicates the type of distribution on transactions created by Oracle Payables or Oracle Purchasing.

**Validation:** This column is system assigned and for internal use only. If the transaction was created by Oracle Payables, then the column holds AP_INVOICE_DISTRIBUTIONS_ALL.LINE_TYPE_LOOKUP_CODE. If the transaction was created by Oracle Purchasing, then the column holds RCV_TRANSACTIONS.TRANSACTION_TYPE.

Destination: PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_DISTRIBUTION_TYPE

**SI_ASSETS_ADDITION_FLAG**

The *supplier invoice assets addition flag* is a system-assigned attribute that indicates the status of the transaction in relation to Oracle Assets.

**Validation:** The column is system assigned and for internal use only. It only applies to supplier cost transactions created in Oracle Payables. The valid values for this column are as follows:

• Y - The transaction has been interfaced to Oracle Assets.

• N - The transaction is not eligible to be interfaced to Oracle Assets.

• T - The transaction is pending interface to Oracle Assets.

Destination: PA_COST_DISTRIBUTION_LINES_ALL.SI_ASSETS_ADDITION_FLAG

**CDL_SYSTEM_REFERENCE5**

A *cost distribution line system reference* is a reference to a record in an external system.
Validation: For transactions created by Oracle Payables, the column holds AP_INVOICE_DISTRIBUTIONS_ALL.INVOICE_DISTRIBUTION_ID. For transactions associated with other transaction sources that have the Raw Cost GL Accounted option enabled, a value is optional. For transactions associated with transaction sources that do not have the Raw Cost GL Accounted option enabled, if you provide a value, then it is ignored.

Destination: COST_DISTRIBUTION_LINES_ALL.SYSTEM_REFERENCE5

When the transaction is created by Oracle Payables, it is also copied to PA_EXPENDITURE_ITEMS_ALL.DOCUMENT_DISTRIBUTION_ID.

**SC_XFER_CODE**

A supplier cost transfer code is a system-assigned attribute that indicates the General Ledger transfer status code of the transaction in Oracle Payables or Oracle Purchasing.

Validation: This column is system assigned and for internal use only. It only applies to transactions created by Oracle Payables or Oracle Purchasing.

Destination: PA_COST_DISTRIBUTION_LINES_ALL.TRANSFER_STATUS_CODE

**ADJUSTED_TXN_INTERFACE_ID**

An adjusted transaction interface identifier is a system-assigned attribute that contains a reference to the transaction interface identifier of an adjusted transaction.

Validation: This column is system assigned and for internal use only. The column only applies to supplier cost transactions created by Oracle Payables or Oracle Purchasing. The column is not used for labor or usage transactions.

Destination: None

**NET_ZERO_ADJUSTMENT_FLAG**

A net zero adjustment flag is a system-assigned attribute that indicates the transaction does not have an impact on the cost amount (in other words, a net-zero value).

Validation: This column is system assigned and is for internal use only. The column is only used for transactions created by Oracle Payables or Oracle Purchasing.

Destination: PA_EXPENDITURE_ITEMS_ALL.NET_ZERO_ADJUSTMENT_FLAG

**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 list:

- **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.

You can get a complete listing of all the rejection reasons from PA_LOOKUPS under the lookup type TRANSACTION REJECTION REASON.

**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, then Oracle Projects rejects the entire expenditure and updates each expenditure item in the expenditure with a status of R (Rejected). However, only the 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. 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.

**Note**: The transaction import validation logic is different when you run the process PRC: Interface Supplier Costs to import transactions from Oracle Purchasing and Oracle Payables. The processes uses predefined supplier cost transaction sources to import expenditure items and it rejects only the expenditure items that fail validation. The process imports the valid expenditure items in the expenditure. You can use the
Review Transactions window to change the date for a rejected expenditure item. Oracle Projects picks up the revised date for the rejected transaction the next time that you run the process PRC: Interface Supplier Costs.

**Examples of Transaction Exceptions**

The following table shows three transactions before Transaction Import:

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Status</th>
<th>Reason</th>
<th>Expenditure ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>2</td>
<td>P</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
<tr>
<td>3</td>
<td>P</td>
<td>[blank]</td>
<td>[blank]</td>
</tr>
</tbody>
</table>

The following table shows the same three transactions after Transaction Import. Only Transaction 1 is shown in the exception report.

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Status</th>
<th>Reason</th>
<th>Expenditure ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P</td>
<td>INVALID PROJECT</td>
<td>1009</td>
</tr>
<tr>
<td>2</td>
<td>P</td>
<td>[blank]</td>
<td>1009</td>
</tr>
<tr>
<td>3</td>
<td>P</td>
<td>[blank]</td>
<td>1009</td>
</tr>
</tbody>
</table>

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 13-56.

- **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.

Review Transactions Window: Expenditure Item Dates for Supplier Costs

The process PRC: Interface Supplier Costs validates expenditure item dates for supplier costs that you interface from Oracle Purchasing and Oracle Payables. If the expenditure item date for an expenditure item fails validation, then the process rejects the transaction and leaves it in the Oracle Projects interface table. You must either change the date setup in Oracle Projects or change the date for the expenditure item. You can use the Review Transactions window to change the date for a rejected expenditure item. Oracle Projects picks up the revised date for the rejected transaction the next time that you run the process PRC: Interface Supplier Costs.

To update the expenditure item date in the Review Transactions window, the Allow Interface Modifications option must be enabled for the transaction source. See: Transaction Sources, Oracle Projects Implementation 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.

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 13-55 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.
1. 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)
A

accumulated period actuals, 2-28
ACTION_IN_REC_TYPE, 3-49
ACTION_OUT_REC_TYPE, 3-54
actuals, 2-28, 6-74, 6-75
actuals summarization API, 6-87
    procedures, 6-88
    project amounts, 6-88
actuals summarization procedures
    project amounts, 6-88
    resource amounts, 6-88
    transaction amounts, 6-89
ADD_ASSET_ASSIGNMENT, 4-4
ADD_BUDGET_LINE, 6-16
ADD_FUNDING, 5-7
ADD_PROJECT_ASSET, 4-2
ADD_RESOURCE_LIST_MEMBER, 3-112
ADD_TASK, 3-69
agreement
    check delete, 5-12
    clear, 5-12
    create, 5-3, 5-10
    delete, 5-4
    initiate, 5-9
    load, 5-9
    update, 5-5, 5-11
agreement and funding
    APIs, 5-1
    agreements
        examples, 5-19, 5-24
        overview, 5-15
agreements and funding, 5-2
    views, 5-2
agreements api
    creat baseline budget, 5-6
allocations
    extensions
        basis, 9-36
        dependencies, 9-38
        descriptive flexfield, 9-37
        offset projects and tasks, 9-35
        offset tasks, 9-34
        source, 9-30
        target, 9-32
API procedures
    planning resource lists, 3-123
apis
    dependency, 3-155
    user authentication, 2-8
APIs
    agreement and funding, 5-1
        procedure definitions, 5-3
    agreement and funding APIs
        security, 5-1
agreement APIs
    collecting agreement information, 5-16
    connecting to an Oracle database, 5-16
    finishing the Load-Execute-Fetch process, 5-18
    getting return values, 5-18
    interfacing agreement information to the server, 5-16
    interfacing funding information to the server, 5-17

Index-2

retrieving error messages, 5-18
starting the server-side process, 5-17
asset, 4-1
budget, 6-7, 6-56
budget APIs
  getting the budget data, 6-57
budget APIs
  connecting to an Oracle database, 6-57
  finishing the Load-Execute-Fetch process, 6-63
  getting budget line data, 6-59
  starting the server-side process, 6-63
common, 2-25
connecting to an Oracle database, 3-54
custom summarization APIs
  budget summarization API, 6-89
custom summarization reporting APIs, 6-87
  actuals summarization API, 6-87
event, 5-27
Event APIs
  API procedure definitions, 5-32
interfacing information to the server, 3-56
introduction, 2-1
Load-Execute-Fetch procedures
  finishing, 3-60
messages, 2-11
Oracle Applications user, 2-6
overview, 1-1, 2-2
overview of APIs, client extensions, and open interfaces, 1-1
planning rates, 6-69
planning resource lists, 3-122
project, 3-2
project APIs
  See check procedures
project definition APIs
  common parameters, 3-7
Project Definition APIs
  procedures, 3-6
project definition APIs, 3-7
project deliverables, 6-1
project performance reporting APIs, 6-90
quick entry fields, 3-55
referencing Oracle Projects entities, 2-24
refresh planning amounts, 6-69
Refresh Planning Amounts API
  Refresh Rates procedure, 6-69
resource, 3-109
resource breakdown structures, 3-140
responsibility, 2-8
security, 5-1
set up a product in Oracle Projects, 2-7
source project, 3-55
source template, 3-55
standard parameters, 2-21
starting the server-side process, 3-59
status, 6-70
Status APIs
  procedures, 6-76
structure APIs, 3-69
Structure APIs
  procedures, 3-68
t APIs
  interfacing budget information to the server, 6-62
task assignments, 3-157
task information
  interfacing to the server, 3-58
tasks
  return values, 3-59
user-defined attributes, 3-85
views, 2-25
where information originates, 2-3
APPLY_LP_PROG_ON_CWV, 3-72
asset
  convert
    CONVERT_PM_ASSETREF_TO_ID, 4-7
create, 4-2
delete, 4-4
fetch
  FETCH_PROJECT_ASSET_ID, 4-7
function definitions, 4-2
load, 4-5
procedure definitions, 4-2
asset assignment
  add, 4-4
delete, 4-4
load, 4-5
asset cost allocation basis extension, 9-39
asset lines
  extension to assign assets to, 9-41
asset lines processing extension, 9-42
assets
  APIs, 4-1
extension, 9-41
functions, 4-2
procedures, 4-2
procedures and views, 4-1
ASSIGN_RBS_TO_PROJECT, 3-153
assignment approval
extensions, 11-1, 11-2
assignment approval notification extension, 11-2
assignment approval workflow
extension, 11-1, 11-2
ASSOCIATE_DLV_TO_TASK, 6-6
ASSOCIATE_DLV_TO_TASK_ASSIGN, 6-6
attribute groups
multi-row, 3-88
single-row, 3-88
automatic events, 10-24

B
BASELINE_BUDGET, 6-20
BASELINE_STRUCTURE, 3-72
baseline budget
api
agreements, 5-6
billing
extensions
concepts, 10-11
debugging, 10-25
deriving cycle dates, 10-2
implementing, 10-6
overview, 10-3
views and procedures, 10-21
surcharges, 10-28
billing extensions
defining, 10-7
budget APIs
record and table datatypes, 6-9
budget lines
adding, 6-16
deleting, 6-33
updating, 6-47, 6-53
budget procedures
definitions, 6-15
budgets
adding lines, 6-16
APIs, 6-7
baseline, 6-20

calculating amounts, 6-21
creating, 6-24
dates
start and end, 6-25
deleting, 6-30, 6-37
deleting lines, 6-33
entry methods, 6-34, 6-58
examples, 6-56, 6-63, 6-66
overview, 6-56
period names, 6-25
procedures, 6-8
procedures and views, 6-7, 6-8
submitting, 6-20
task level of BEM, 6-25
types
predefined, 6-57
updating, 6-39
budgets and forecasts
extensions
budget calculation, 12-3
budget verification, 12-7
budget workflow, 12-8
estimate to complete generation method, 12-9
burdening
cost plus API, 4-7
extension for costing, 9-28
burdening commitments client extension, 12-18
PSI
function, 12-18
same line burdening, 12-18
burden resource extension, 9-29
client column values procedure, 9-30
client grouping function, 9-29
burden transactions
importing, 13-10
business rules
PA_AGREEMENT_PUB.ADD_FUNDING, 5-7
PA_AGREEMENT_PUB.CHECK_ADD_FUNDING_OK, 5-13
PA_AGREEMENT_PUB.CHECK_DELETE_AGREEMENT_OK, 5-12
PA_AGREEMENT_PUB.CHECK_DELETE_FUNDING_OK, 5-14
PA_AGREEMENT_PUB.CHECK_UPDATE_FUNDING_OK, 5-14
pa_agreement_pub.create_agreement, 5-3
designing, 10-8
overview, 10-3
views and procedures, 10-21
writing procedures, 10-24
budget calculation, 12-3
designing, 12-3
procedures, 12-5
types of calculations, 12-4
budget verification, 12-7
budget workflow, 12-8
designing, 12-8
procedures, 12-8
burden costing, 9-28
burden resource, 9-29
candidate notification workflow, 11-4
capital event processing, 9-43
procedure, 9-44
capitalized interest, 9-44, 9-45
CIP account override, 9-49
procedure, 9-49
CIP grouping, 9-46
example, 9-47
commitment changes
package, 12-19
control item document numbering, 12-10
procedures, 12-10
cost accrual billing, 10-33
cost accrual identification, 10-34
cross charge, 9-51
custom performance measure, 12-13
procedures, 12-13
custom performance measure
package, 12-13
depreciation account override, 9-50
procedure, 9-50
depreciation expense account override, 9-50
descriptive flexfield mapping, 8-14
description, 8-14
designing
logic, 7-2
estimate to complete generation method, 12-9
procedures, 12-9
funding revaluation factor, 10-1
implementing
SectHead, 7-2
issue and change workflow, 12-11
procedures, 12-11
labor billing, 10-36
designing, 10-36
writing, 10-35
labor costing, 9-14
designing, 9-14
processing, 9-14
writing, 9-14
labor transaction, 9-16
designing, 9-16
frequently asked questions, 9-22
processing, 9-16
related transactions, 9-16
writing, 9-16
non-labor billing, 10-38
designing, 10-38
writing, 10-39
output tax, 10-42
overtime calculation, 9-24
designing, 9-24
report, 9-26
writing, 9-26
overview, 1-2
overview and list
SectHead, 7-1
performance status, 12-14
procedure, 12-14
performance status
package, 12-14
progress management, 12-2
project and task date, 8-5
project management, 12-2
project security, 8-1
project status inquiry, 12-15
Project Status Inquiry
burdening commitments, 12-18
commitment changes, 12-19
project status inquiry burdening commitments, 12-18
project status inquiry commitment changes, 12-19
project status report workflow, 12-12
procedures, 12-12
project verification, 8-3
designing, 8-3
processing, 8-3
writing, 8-3
project workflow, 8-8
processing, 8-8
writing, 8-8
PSI, 12-15
procedures, 12-15
receivables installation override, 10-43
retention billing
writing, 10-40
syntax, 7-6
task date
customize dates procedure, 8-5
transaction control, 9-1
designing, 9-2
error messages, 9-3
frequently asked questions, 9-7
parameters, 9-4
processing, 9-2
validation, 9-2
writing, 9-3
transaction import, 8-10
verify organization change, 8-9
workplan workflow, 12-1
commitment changes client extension, 12-19
procedure, 12-19
commitments, 6-74
extension for burdening, 12-18
extension for tracking changes, 12-19
common APIs, 2-25
composite datatypes, 2-22, 5-24, 6-66
record and table datatypes, 3-22, 3-124, 3-142, 6-77
composite data types
record and table datatypes, 3-158
cost plus API, 4-7
example, 4-9
cross charge
extension for
internal payables invoice attributes
override, 9-58
extensions for
cost accrual ID, 10-34
determining transfer price, 9-54
overriding processing method, 9-52
overriding providers and receivers, 9-51
overriding transfer price, 9-55
overriding transfer price currency, 9-57
currencies
extensions
transfer price currency, 9-57
importing transactions in foreign, 13-12
rounding, 13-14
CUSTOMER_TBL_TYPE, 3-45
D
datatatypes
BUDGET_LINE_IN_REC_TYPE, 6-10
BUDGET_LINE_OUT_REC_TYPE, 6-13
CALC_BUDGET_LINE_OUT_REC_TYPE, 6-13
composite, 2-22, 5-24, 6-66
for planning resource lists, 3-124
record and table, 3-22, 3-142, 3-158, 6-77
standard, 2-31
dates for
budgets, 6-25
projects, 3-8, 3-13, 8-5
tasks, 3-8, 3-16, 8-5
defaults
initialization, 2-27
DELETE_AGREEMENT, 5-4
DELETE.Asset_Assignment, 4-4
DELETE.BASELINE_BUDGET, 6-30
DELETE.BUDGET_LINE, 6-33
DELETE.DELIVERABLE_ACTIONS, 6-6
DELETE.DELIVERABLES, 6-6
DELETE.DLV_TO_TASK_ASSCN, 6-7
DELETE.DLV_TO_TASK_ASSIGN, 6-7
DELETE.DRAFT_BUDGET, 6-37
DELETE.EVENT, 5-32
DELETE.FUNDING, 5-8
DELETE_PLAN_RL_FORMAT, 3-135
business rules, 3-135  
parameters, 3-135  
DELETE_PLANNING_RESOURCE, 3-134  
business rules, 3-134  
parameters, 3-134  
DELETE_PROJECT, 3-9  
DELETE_PROJECT_ASSET, 4-4  
DELETE_RESOURCE_LIST, 3-115, 3-133  
DELETE_RESOURCE_LIST_MEMBER, 3-116  
DELETE_STRUCTURE_VERSION, 3-74  
DELETE_TASK, 3-74  
DELIVERABLE_IN_REC_TYPE, 3-47  
DELIVERABLE_OUT_REC_TYPE, 3-48  
dependency  
  apis, 3-155  
  views, 3-155, 3-155  
dependency procedures, 3-155  
  create_dependency, 3-155  
  delete_dependency, 3-157  
  update_procedures, 3-156  
depreciation expense account override extension, 9-50  
descriptive flexfields  
  allocation, 9-37  
  mapping extension, 8-14  
examples  
  composite datatypes, 5-24, 6-66  
  integrating data, 3-54, 5-15, 6-56  
  Load-Execute-Fetch procedures, 3-60, 5-19, 6-63  
  restricting actions, 2-30  
  tracking dates, 8-5  
EXEC_CREATE_RBS, 3-152  
EXEC_CREATE_RESOURCE_LIST, 3-118, 3-135  
EXEC_UPDATE_RBS, 3-152  
EXEC_UPDATE_RESOURCE_LIST, 3-118, 3-136  
EXECUTE_CALCULATE_AMOUNTS, 6-52  
EXECUTE_CREATE_AGREEMENT, 5-10  
EXECUTE_CREATE_DRAFT_BUDGET, 6-53  
EXECUTE_CREATE_DRAFT_FINPLAN, 6-53  
EXECUTE_CREATE_EVENT, 4-6, 5-33  
EXECUTE_CREATE_PROJECT, 3-17  
EXECUTE_UPDATE_AGREEMENT, 5-11  
EXECUTE_UPDATE_BUDGET, 6-54  
EXECUTE_UPDATE_EVENT, 5-34  
EXECUTE_UPDATE_PROJECT, 3-18  
EXECUTE_UPDATE_TASK_PROGRESS, 6-86  
expenditures, 3-110, 3-110  
  approving, 9-11  
expense reports  
  approving, 9-11  
  importing, 13-17  
external systems  
  overview, 2-2  
error messages  
  creation, 2-10  
  retrieving, 2-25, 3-59, 6-63  
event  
  check delete, 5-34  
  clear, 5-34  
  create, 4-6, 5-32, 5-33  
  delete, 5-32  
  fetch, 5-34  
  initiate, 5-33  
  load, 5-33  
  update, 4-3, 5-32, 5-34  
event apis  
  package variables, 5-28  
events  
  APIs, 5-27  
  automatic, 10-24  
  procedures, 5-27  
event types, 3-110  
E  
FETCH_BUDGET_LINE, 6-55  
FETCH_CALCULATE_AMOUNTS, 6-55  
FETCH_EVENT, 5-34  
FETCH_FUNDING, 5-12  
FETCH_MEMBERS, 3-118, 3-137  
FETCH_PLAN_FORMAT, 3-136  
FETCH_RBS_ELEMENT, 3-152  
FETCH_RBS_HEADER, 3-151  
FETCH_RBS_VERSION, 3-151  
FETCH_RESOURCE_LIST, 3-119, 3-136  
FETCH_STRUCTURE_VERSION, 3-80
FETCH_TASK, 3-80
FETCH_TASK_VER, 3-81
FETCH_TASKS, 3-80
files
  output
    paamgcrole.lst, 2-5
    paamgcuser.lst, 2-6
FREEZE_RBS_VERSION, 3-153
funding
  add, 5-7
  check adding, 5-13
  check deleting, 5-14
  check update, 5-14
  create, 5-7
  delete, 5-8
  fetch, 5-12
  load, 5-10
  update, 5-8

G
GET_ACCUM_PERIOD_INFO, 2-28
GET_DEFAULTS, 2-27
GET_DELETED_TASKS_FROM_OP, 3-76
GET_MESSAGES, 2-25
GET_PROJECT_ID, 6-39
GET_TASK_VERSION, 3-75
Get Burden Amount, 4-8
global variables, 2-9
grants, 2-7
graphics
  See illustrations

H
hard limits, 10-24

I
importing
  manufacturing costs, 13-10
  transactions, 13-1
importing database, 2-7
INIT_AGREEMENT, 5-9
INIT_BUDGET, 6-55
INIT_CALCULATE_AMOUNTS, 6-55
INIT_CREATE_RESOURCE_LIST, 3-119, 3-137
INIT_EVENT, 5-33
INIT_project, 3-19
INIT_RBS_PROCESSING, 3-150
INIT_UPDATE_MEMBERS, 3-119, 3-137
INIT_UPDATE_TASK_PROGRESS, 6-87
integration
  examples, 3-54, 5-15, 6-56
  with other systems, 2-2
  with project management systems, 2-5
interface tables, 13-27
internal payables invoice attribute override
  extension
    sectHead, 9-58
invoices
  approving, 10-40
  extension, 10-40
  releasing, 10-40
issue management
  client extensions
    control item document numbering, 12-10
    issue and change workflow, 12-11

L
labor
  extensions
    billing, 10-35
    costing, 9-14
    transaction, 9-16
labor costs
  extensions, 9-14
LOAD_ACTION, 6-3
LOAD_ACTIONS, 6-4
LOAD_AGREEMENT, 5-9
LOAD_ASSET_ASSIGNMENT, 4-5
LOAD_BUDGET_LINE, 6-55
LOAD_CLASS_CATEGORY, 3-19
LOAD_DELIVERABLE, 6-3
LOAD_DELIVERABLES, 6-3
LOAD_EVENT, 5-33
LOAD_EXTENSIBLE_ATTRIBUTE, 3-86
  example, 3-91
LOAD_EXTENSIBLE_ATTRIBUTES, 3-87
  example, 3-99
LOAD_FUNDING, 5-10
LOAD_KEY_MEMBER, 3-20
LOAD_MEMBERS, 3-119
LOAD_ORG_ROLE, 3-20
LOAD_PLANNING_RESOURCE, 3-139
LOAD_PROJECT, 3-20
LOAD_PROJECT_ASSET, 4-5
LOAD_RBS_ELEMENTS, 3-150
LOAD_RBS_HEADER, 3-150
LOAD_RBS_VERSION, 3-150
LOAD_RESOURCE_FORMAT, 3-139
LOAD_RESOURCE_INFO, 6-56
LOAD_RESOURCE_LIST, 3-120, 3-137
LOAD_STRUCTURE, 3-81
LOAD_TASK, 3-81, 3-83
LOAD_TASK_PROGRESS, 6-86
parameters, 6-79
Load-Execute-Fetch procedures
budgets, 6-9
examples, 3-60, 5-19, 6-63
overview, 2-31
projects, 3-6
resources, 3-111
status, 6-76
structure, 3-68

M
manufacturing costs
importing, 13-10
messages, 2-10
displaying, 2-11
handling, 2-10
list, 2-11

N
named notation, 2-25
names
user, 2-8
naming projects, 3-11
non-labor
extensions
billing, 10-38
numbering
projects, 3-7
tasks, 3-76

O
open interfaces
overview, 1-2

transaction import, 13-2
Oracle Projects APIs
overview, 2-2
Oracle Receivables
extensions to
determine transaction type, 10-45
override installation, 10-43
integrating with, 10-45
Oracle Workflow
extensions for
budget workflow, 12-8
Project Resource Management, 11-1, 11-2
project status report workflow, 12-12
project verification, 8-3
project workflow, 8-8
workplan workflow, 12-1
organizations
extension to verify changes, 8-9
output files
paamgrole.lst, 2-5
paamgcuser.lst, 2-6
overtime
calculating
extension for, 9-24
overviews
client extensions
sectHead, 7-1
Transaction Import, 13-1

PA_ACCUM_CMT_TXNS_V, 6-74
PA_ACCUM_RSRC_ACT_V, 6-74
PA_ACCUM_RSRC_CMT_V, 6-74
PA_ACCUM_RSRC_COST_BGT_V, 6-75
PA_ACCUM_RSRC_REV_BGT_V, 6-75
PA_ACCUM_WBS_ACT_V, 6-75
PA_ACCUM_WBS_CMT_V, 6-75
PA_ACCUM_WBS_COST_BGT_V, 6-75
PA_ACCUM_WBS_REV_BGT_V, 6-75
PA_ACT_BY_GL_PERIOD_V, 6-75
PA_ACT_BY_PA_PERIOD_V, 6-75
PA_AGREEMENT_PUB.UPDATE_AGREEMENT_T, 5-5
PA_AGREEMENT_TYPE_LOV_V, 5-2
PA_AMG_RESOURCE_INFO_V, 3-110
PA_ASSET_BOOKS_LOV_V, 4-1
procedure definitions
   | budget, 6-8
   | check project, 3-6
   | check structure and task, 3-68
   | dependency, 3-155
   | Load-Execute-Fetch, 3-6, 3-68, 3-111, 6-9, 6-76
   | overview, 2-31
   | project, 3-6
   | resource breakdown structures, 3-141
   | resource list, 3-111, 3-111
   | status, 6-76, 6-83
   | structure and task, 3-68
   | task assignments, 3-158
product codes, 6-24, 6-27
prohibiting actions, 2-30
PROJECT_IN_REC_TYPE datatype, 3-22
PROJECT_OUT_REC_TYPE datatype, 3-33
PROJECT_ROLE_TBL_TYPE datatype, 3-34
project deliverable APIs, 6-1
project deliverables APIs
   | procedures, 6-2
   | views, 6-2
projects
   | APIs, 3-2
   | attributes
      | changing, 3-11
   | commitments, 6-74
   | creating, 3-8
   | dates
      | start and finish, 3-8, 3-13, 8-6
   | defining, 3-6
   | deleting, 3-9
   | examples, 3-60
   | numbering, 3-7
   | overview, 3-54
   | procedures and views, 3-6
   | updating, 3-10
project security extension, 8-1
   | logic, 8-1
   | parameters, 8-1
   | writing, 8-1
project status
   | extensions, 8-3, 8-8
Project Status Inquiry, 12-15, 12-18
   | adding columns, 12-15
   | client extension, 12-15
      | burdening commitments, 12-18
commitment changes, 12-19
project status reports
   | extensions, 12-12
   | using the project status report workflow extension, 12-12
project workflow
   | extension to call, 8-3
   | extension to change status, 8-8
PSI, 2-4, 6-70
   | client extension, 12-15, 12-18
R
record and table datatypes, 3-22, 6-77
   | budget, 6-9
   | for planning resource list APIs, 3-124
   | for resource breakdown structure APIs, 3-142
   | for task assignment APIs, 3-158
refresh planning amounts
   | API, 6-69
refresh rates
   | refresh planning rates and amounts API, 6-69
reports and listings
   | Pre-Approved Expenditures Entry Audit Report, 13-23
resource breakdown structure
   | fetching elements for, 3-152
   | fetching versions for, 3-151
   | freezing resource breakdown structure versions, 3-153
   | initializing processing, 3-150
   | loading versions for, 3-150
resource breakdown structures
   | API definitions, 3-147
   | APIs, 3-140
   | assigning to projects, 3-153
   | copying a working version, 3-148
   | creating, 3-147
   | fetching headers for, 3-151
   | loading elements for, 3-150
   | loading headers for, 3-150
   | updating, 3-149
   | views, 3-140, 3-141
resource list members
   | adding, 3-112
   | deleting, 3-116, 3-116
   | retrieving, 3-110
sorting, 3-116
updating, 3-117
resource lists
creating, 3-113, 3-131
deleting, 3-115, 3-115, 3-133
names, 3-120, 3-137
retrieving, 3-110, 3-110, 3-110
updating, 3-112, 3-114, 3-117, 3-132
resource procedures
definitions, 3-112
resources
APIs, 3-109
commitments, 6-74
defining, 3-110, 3-110, 3-110, 3-110, 3-111
retrieving, 3-110
tracked as labor, 6-71
values predefined by Oracle Projects, 6-71
vendors, 3-111
views, 3-109, 3-111
restricting actions, 2-30
retention
billing extension, 10-40
retrieving error messages, 2-25, 3-59, 6-63
revenue categories, 3-111

S
scripts
paamgcrole.sql, 2-5
paamguser.sql, 2-6
paamggran.sql, 2-7
security
project
client extension, 8-1
security requirements, 2-8
SET_GLOBAL_INFO, 2-9
SET_PROJECT_ID, 6-39
SORT_RESOURCE_LIST_MEMBERS, 3-116
standard API parameters, 2-21
standard datatypes, 2-31, 3-60, 5-19
status
defining, 6-76
procedures, 6-76
views, 6-74
status APIs
parameters, 6-79
status APIs, 6-70
structure
defining, 3-68
procedures and views, 3-68
Structure APIs
structures, 3-66
tasks, 3-66
submitting
budgets, 6-20
summary amounts
commitment, 6-75
cost budget, 6-75
resource commitment, 6-74
resource cost budget, 6-75
resource revenue, 6-74
resource revenue budget, 6-75
revenue, 6-75
revenue budget, 6-75
systems
external, 2-2

T
TABLE_IN_TBL_TYPE datatype, 3-35
table and record datatypes, 3-22, 6-77
for planning resource list APIs, 3-124
for resource breakdown structure APIs, 3-142
for task assignment APIs, 3-158
TASK_OUT_TBL_TYPE datatype, 3-42
task assignment
update_task_assignment_periods, 3-165
task assignment procedures, 3-162
execute_create_task_asgmts, 3-163
execute_update_task_asgmts, 3-164
load_task_assignment_periods, 3-163
load_task_assignments, 3-163
task assignments
APIs, 3-157
create_task_assignment_periods, 3-164
create_task_assignments, 3-164
delete_task_assignments, 3-166
fetch_task_assignments, 3-166
init_task_assignments, 3-167
update_task_assignments, 3-165
views, 3-157, 3-158
task assignments
convert_pm_taref_to_id, 3-167
tasks
  adding, 3-69
attributes
  changing, 3-15
commitments, 6-74
dates
  start and finish, 3-8, 3-16, 8-5, 8-6
deleting, 3-10, 3-74
fields, 3-16
interface to Oracle Projects, 3-15
loading task information, 3-81
moving within the WBS, 3-16
numbering, 3-15
subtasks
  creating, 3-69
  updating, 3-76
taxes
  classification codes
    defaults, 10-42
templates
  for client extensions, 7-6
transaction controls
  case studies, 9-7, 9-9, 9-10
  extensions, 9-1
transaction import
  defining sources, 13-6
  examples, 13-19, 13-21
  exceptions, 13-53
  expenditure type classes for, 13-7
  grouping transactions, 13-16
interface tables, 13-25
  column descriptions, 13-28
  expenditure, 13-26
  populating, 13-4, 13-27
loading
  accounted/unaccounted items, 13-9
  burden transactions, 13-10
  costed/uncosted items, 13-8
  foreign currency transactions, 13-12
  manufacturing costs, 13-10
options, 13-15
overview, 13-1, 13-4
process diagram, 13-2
process flow diagram, 13-2
rejections
  codes for, 13-54
  correcting, 13-56
reporting, 13-6
reviewing transactions, 13-55
rounding limit, 13-14
tables
  interface control, 13-28
transactions
  adjusting, 13-24
  importing, 13-6
  purging, 13-24
  viewing, 13-23
unmatched negative transactions for, 13-7, 13-8
validation
  H:1, 13-25
Transaction Import Client Extensions, 8-10
transactions
  adjusting, 9-22
  unmatched negative, 13-7, 13-8
transaction types
  extension, 10-45

U

UPDATE_BUDGET, 6-39
UPDATE_BUDGET_LINE, 6-47
UPDATE_DELIVERABLE, 6-4
UPDATE_DELIVERABLE_ACTION, 6-5
UPDATE_EARNED_VALUE, 6-83
UPDATE_EVENT, 4-3, 5-32
UPDATE_FUNDING, 5-8
UPDATE_PROGRESS, 6-84
  parameters, 6-79
UPDATE_PROJECT, 3-10
UPDATE_RBS, 3-149
UPDATE_RESOURCE_LIST, 3-114, 3-132
UPDATE_RESOURCE_LIST_MEMBER, 3-117
UPDATE_TASK, 3-76
User-Defined Attribute API
  example, 3-87
user-defined attributes
  APIs, 3-85
  loading, 3-86, 3-87
  procedures, 3-85
usernames, 2-8

V

variables
global, 2-9
vendors, 3-111
verifying data, 3-2, 3-85
view definitions
  asset, 4-1
  budget, 6-7
  dependency, 3-155, 3-155
  planning resource lists, 3-122
  project, 3-2
  resource breakdown structures, 3-140, 3-141
  resources, 3-109, 3-111
  status, 6-71, 6-74
  structure, 3-67
  task assignments, 3-158
views
  task assignments, 3-157

W
WBS, 3-2, 3-16
window illustrations
  Review Transactions, 13-53
workplan and progress management
  extensions for progress management, 12-2
workplan workflow extension, 12-1