Oracle® Process Manufacturing

Cost Management API User's Guide

Release 11*i*

Part No. A95270-03

August 2004

ORACLE®

Oracle Process Manufacturing Cost Management API User's Guide, Release 11*i*

Part No. A95270-03

Primary Authors: Michele-Andrea Fields

Contributors: Uday Moogala, Rajesh Seshadri

Contents

2 Cost Management API Usage

3 Technical Overview

4 Business Objects

A Messages and Errors

Index

Send Us Your Comments

Oracle Process Manufacturing Cost Management API User's Guide, Release 11*i*

Part No. A95270-03

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

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

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

- Electronic mail: appsdoc_us@oracle.com
- FAX: (650) 506-7200 Attn: Oracle Applications Documentation Manager
- Postal service:
  Oracle Corporation
  Oracle Applications Documentation Manager
  500 Oracle Parkway
  Redwood Shores, CA 94065
  USA

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

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

Preface

Welcome to the Oracle Process Manufacturing Cost Management API User's Guide, Release 11*i*.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle Process Manufacturing.

  If you have never used Oracle Process Manufacturing, Oracle suggests you attend one or more of the Oracle Applications training classes available through Oracle University.
- Oracle Self-Service Web Applications.

  To learn more about Oracle Self-Service Web Applications, read the *Oracle Self-Service Web Applications Implementation Manual*.
- The Oracle Applications graphical user interface.

  To learn more about the Oracle Applications graphical user interface, read the *Oracle Applications User's Guide*.

See Other Information Sources for more information about Oracle Applications product information.

How To Use This Guide

The Oracle Process Manufacturing Cost Management API User's Guide contains the information you need to understand and use Oracle Process Manufacturing. This guide contains four chapters:

- Chapter 1 describes the Application Program Interfaces (APIs) that support external interfaces to the OPM Cost Management tables.
- Chapter 2 provides information on the usage and layout of the Cost Management APIs.
- Chapter 3 provides the relationships between Costing API table structure and its entities. Discusses Costing API business objects, the entity relationship diagram, business object interface design, creating a cost, and importing cost data structures.
- Chapter 4 provides details on each Cost Managment API.
- Appendix A provides message handling, the interpretation of error conditions, and an understanding of error messages.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Other Information Sources

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

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

Online Documentation

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

- **PDF Documentation-** See the Online Documentation CD for current PDF documentation for your product with each release. This Documentation CD is also available on Oracle*MetaLink* and is updated frequently.
- **Online Help -** You can refer to Oracle Applications Help for current HTML online help for your product. Oracle provides patchable online help, which you can apply to your system for updated implementation and end user documentation. No system downtime is required to apply online help.
- **Release Content Document -** See the Release Content Document for descriptions of new features available by release. The Release Content Document is available on Oracle*MetaLink*.
- **About document -** Refer to the About document for information about your release, including feature updates, installation information, and new documentation or documentation patches that you can download. The About document is available on Oracle*MetaLink*.

Related Guides

Oracle Process Manufacturing shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other guides when you set up and use Oracle Process Manufacturing.

You can read the guides online by choosing Library from the expandable menu on your HTML help window, by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle Store at http://oraclestore.oracle.com.

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). This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

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

Guides Related to This Product

Accounting Setup User's Guide

The OPM Accounting Setup application is where users set up global accounting attributes about the way financial data will be collected by OPM. These attributes include such things as account keys, financial calendars, and account segments. Since OPM is closely integrated with Oracle General Ledger (GL), much of the attributes are defined in the Oracle GL instead of OPM, and therefore, the windows are display only within OPM. The *Oracle Process Manufacturing Accounting Setup User's Guide* describes how to setup and use this application.

Cost Management User's Guide

The OPM Cost Management application is used by cost accountants to capture and review the manufacturing costs incurred in their process manufacturing businesses. The *Oracle Process Manufacturing Cost Management User's Guide* describes how to setup and use this application.

Manufacturing Accounting Controller User's Guide

The Manufacturing Accounting Controller application is where users define the impact of manufacturing events on financials. For example, event RCPT (Inventory Receipts) results in a debit to inventory, a credit to accrued accounts payable, a debit or a credit to purchase price variance, etc. These impacts are predefined in the Manufacturing Accounting Controller application so users may begin using OPM to collect financial data out-of-the-box, however, they may also be adjusted per your business needs. The *Oracle Process Manufacturing Manufacturing Accounting Controller User's Guide* describes how to setup and use this application.

Oracle Financials Integration User's Guide

Since OPM is closely integrated with Oracle General Ledger, financial data that is collected about the manufacturing processes must be transferred to the Oracle Financials applications. The OPM Oracle Financials Integration application is where users define how that data is transferred. For example, users define whether data is transferred real time or batched and transferred at intervals. The *Oracle Process Manufacturing Oracle Financials Integration User's Guide* describes how to setup and use this application.

Inventory Management User's Guide

The OPM Inventory Management application is where data about the items purchased for, consumed during, and created as a result of the manufacturing process are tracked. The *Oracle Process Manufacturing Inventory Management User's Guide* includes information to help you effectively work with the Oracle Process Manufacturing Inventory application.

Physical Inventory User's Guide

Performing physical inventory count is the most accurate way to get an accounting of all material quantities purchased, manufactured, and sold, and update your onhand quantities accordingly. The OPM Physical Inventory application automates and enables the physical inventory process. The *Oracle Process Manufacturing Physical Inventory User's Guide* describes how to setup and use this application.

Order Fulfillment User's Guide

The OPM Order Fulfillment application automates sales order entry to reduce order cycle time. Order Fulfillment enables order entry personnel to inform customers of scheduled delivery dates and pricing. The *Oracle Process Manufacturing Order Fulfillment User's Guide* describes how to setup and use this application.

Purchase Management User's Guide

OPM Purchase Management and Oracle Purchasing combine to provide an integrated solution for Process Manufacturing. Purchase orders are entered in Oracle Purchasing and received in OPM. Then, the receipts entered in OPM are sent to Oracle Purchasing. The *Oracle Process Manufacturing Purchase Management User's Guide* describes how to setup and use this integrated solution.

Using Oracle Order Management with Process Inventory Guide

Oracle Process Manufacturing and Oracle Order Management combine to provide an integrated solution for process manufacturers. The manufacturing process is tracked and handled within Oracle Process Manufacturing, while sales orders are taken and tracked in Oracle Order Management. Process attributes, such as dual UOM and lot control, are enabled depending on the inventory organization for the item on the sales order. Order Management accepts orders entered through Oracle Customer Relationship Management (CRM). Within CRM, orders can originate from TeleSales, Sales Online, and iStore, and are booked in Order Management, making the CRM suite of products available to Process customers, through Order Management. The *Oracle Order Management User's Guide* and *Using Oracle Order Management with Process Inventory Guide* describes how to setup and use this integrated solution.

Process Execution User's Guide

The OPM Process Execution application lets you track firm planned orders and production batches from incoming materials through finished goods. Seamlessly integrated to the Product Development application, Process Execution lets you convert firm planned orders to single or multiple production batches, allocate ingredients, record actual ingredient usage, and then complete and close production batches. Production inquiries and preformatted reports help you optimize inventory costs while maintaining a high level of customer satisfaction with on-time delivery of high quality products. The *OPM Process Execution User's Guide* presents overviews of the tasks and responsibilities for the Production Supervisor and the Production Operator. It provides prerequisite setup in other applications, and details the windows, features, and functionality of the OPM Process Execution application.

Using Oracle Advanced Planning and Scheduling with Oracle Process Manufacturing

Oracle Process Manufacturing and Oracle Advanced Planning and Scheduling (APS) combine to provide a solution for process manufacturers that can help increase planning efficiency. This solution provides for constraint-based planning, performance management, materials management by exception, mixed mode manufacturing that enables you to choose the best method to produce each of your products, and combine all of these methods within the same plant/company. The *Using Oracle Advanced Planning and Scheduling with Oracle Process Manufacturing User's Guide* describes how to setup and use this application.

MPS/MRP and Forecasting User's Guide

The Oracle Process Manufacturing Material Requirements Planning (MRP) application provides long-term "views" of material demands and projected supply actions to satisfy those demands. The Master Production Scheduling (MPS) application lets you shorten that view to a much narrower and immediate time horizon, and see the immediate effects of demand and supply actions. The *Oracle Process Manufacturing MPS/MRP and Forecasting User's Guide* describes how to setup and use this application.

Capacity Planning User's Guide

The OPM Capacity Planning User's Guide describes the setup required to use OPM with the Oracle Applications Advanced Supply Chain Planning solutions. In addition, Resource setup, used by the OPM Production Execution and New Product Development applications, is also described.

Using Oracle Process Manufacturing with Oracle Manufacturing Scheduling

Oracle Process Manufacturing integrates with Oracle Manufacturing Scheduling to manage and utilize resources and materials. Through the Process Manufacturing application, you set up manufacturing, inventory, procurement and sales order data. Through the Manufacturing Scheduling application, you can optimize the schedule based on resource and component constraints and user predefined priorities. Using different optimization objectives, you can tailor Manufacturing Scheduling to meet your needs.

Using Oracle Manufacturing Scheduling helps you improve productivity and efficiency on your shop floor. By optimally scheduling shop floor jobs, and being able to quickly react to unplanned constraints, you can lower manufacturing costs, increase resource utilization and efficiency, and increase customer satisfaction through improved on-time delivery. The *Using Oracle Process Manufacturing with Oracle Manufacturing Scheduling User's Guide* describes how to setup and use this integrated solution.

Product Development User's Guide

The Oracle Process Manufacturing Product Development application provides features to manage formula and laboratory work within the process manufacturing operation. It lets you manage multiple laboratory organizations and support varying product lines throughout the organization. You can characterize and simulate the technical properties of ingredients and their effects on formulas. You can optimize formulations before beginning expensive laboratory test batches. Product Development coordinates each development function and enables a rapid,

enterprise-wide implementation of new products in your plants. The *Oracle Process Manufacturing Product Development User's Guide* describes how to setup and use this application.

Quality Management User's Guide

The Oracle Process Manufacturing Quality Management application provides features to test material sampled from inventory, production, or receipts from external suppliers. The application lets you enter specifications and control their use throughout the enterprise. Customized workflows and electronic record keeping automate plans for sampling, testing, and result processing. You can compare specifications to assist in regrading items, and match customer specifications. Aggregate test results and print statistical assessments on quality certificates. Several preformatted reports and inquiries help manage quality testing and reporting. The *Oracle Process Manufacturing Quality Management User's Guide* describes how to set up and use this application.

Implementation Guide

The *Oracle Process Manufacturing Implementation Guide* offers information on setup. That is, those tasks you must complete following the initial installation of the Oracle Process Manufacturing software. Any tasks that must be completed in order to use the system out-of-the-box are included in this manual.

System Administration User's Guide

Much of the System Administration duties are performed at the Oracle Applications level, and are therefore described in the *Oracle Applications System Administrator's Guide.* The *Oracle Process Manufacturing System Administration User's Guide* provides information on the few tasks that are specific to OPM. It offers information on performing OPM file purge and archive, and maintaining such things as responsibilities, units of measure, and organizations.

API User's Guides

Public Application Programming Interfaces (APIs) are available for use with different areas of the Oracle Process Manufacturing application. APIs make it possible to pass information into and out of the application, bypassing the user interface. Use of these APIs is documented in individual manuals such as the *Oracle Process Manufacturing Inventory API User's Guide, Oracle Process Manufacturing Process Execution API User's Guide, Oracle Process Manufacturing Product Development Formula API User's Guide, Oracle Process Manufacturing Product Development Recipe API User's Guide, Oracle Process Manufacturing Quality Management API User's Guide,*

and the *Oracle Process Manufacturing Cost Management API User's Guide.* Additional API User's Guides are periodically added as additional public APIs are made available.

Installation and System Administration

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind Applications-wide features such as Business Intelligence (BIS), languages and character sets, and Self-Service Web Applications.

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications and the Oracle 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 guides and implementation guides.

Upgrading Oracle Applications

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

"About" Document

For information about implementation and user documentation, instructions for applying patches, new and changed setup steps, and descriptions of software updates, refer to the "About" document for your product. "About" documents are available on Oracle*MetaLink* for most products starting with Release 11.5.8.

Maintaining Oracle Applications

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

Oracle Applications System Administrator's Guide

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

Oracle Alert User's Guide

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

Oracle Applications Developer's Guide

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

Oracle Applications User Interface Standards for Forms-Based Products

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

Other Implementation Documentation

Oracle Applications Product Update Notes

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

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 Process Manufacturing implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications product data. This guide also provides information on creating custom reports on flexfields data.

Oracle eTechnical Reference Manuals

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

Oracle Applications Message Manual

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

Training and Support

Training

Oracle offers a complete set of training courses to help you and your staff master Oracle Process Manufacturing 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 one of our many education centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization structure, terminology, and data as examples in a customized training session delivered at your own facility.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Process Manufacturing 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 Oracle Applications can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

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

About Oracle

Oracle develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 160 software modules for financial management, supply chain management, manufacturing, project systems, human resources and customer relationship management.

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

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

Your Feedback

Thank you for using Oracle Process Manufacturing and this user guide.

Oracle values your comments and feedback. In this guide is a reader's comment form that you can use to explain what you like or dislike about Oracle Process Manufacturing or this user guide. Mail your comments to the following address or call us directly at (650) 506-7000.

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

Or, send electronic mail to appsdoc_us@oracle.com.

1

Cost Management APIs

This document describes the Application Program Interfaces (APIs) that support external interfaces to the Oracle Process Manufacturing (OPM) Cost Management tables. The topics discussed in this topic are:

- Introducing the Cost Management APIs
- Oracle Applications Packages Supplied
- Cost Management API Bill of Materials

Introducing the Cost Management APIs

Cost Management APIs let you to import costing information from an existing cost management system into the Oracle Process Manufacturing Cost Management tables. When you import this information you can now include all pertinent information using a friendly tool that does not have cryptic IDs and system specific information. Cost Management APIs can process virtually all cost types. The interface insures that your imported costs contain the same detail as those you enter manually on the OPM Cost Management window.

What's In This Document?

This document describes the basic business needs, major features, architecture, and components for the Insert, Update, and Delete features for the Cost Management APIs. Much of the application is divided into application-specific objects that let you to link OPM functionality into your own programs. The interfaces can make use of the standard functionality and logic implemented in the Cost Management application.

Cost Management APIs are currently written in PL/SQL which can be called by your own programs. To make use of these APIs, you must code your wrapper function that passes the appropriate parameters to the APIs. Your program is also responsible for connecting to a database before calling an API function, and disconnecting from the database upon return. You can also choose to write log files before calling and after returning from a function. If there is a problem during execution of a call, then the APIs return one of the following status codes:

- S for success
- E for error
- U unknown or unexpected status

Basic Business Needs

These APIs let you have an Application Programming Interface (API) available for the maintenance of cost information for an Item and Resource. The purpose of these APIs is to provide a mechanism, in addition to an entry window, through which you can manipulate cost information in the Oracle Process Manufacturing (OPM) application. The usage of this API is to:

- Help new customers load bulk of item cost, resource cost, burden information, and allocation basis information into OPM from external systems bypassing laborious and error prone manual data entry.

- Enable existing customers to interface OPM with their custom applications.
- Allow a systematic approach to manipulate cost information in OPM without compromising any business rules or database constraints of OPM.

In addition to above, this APIs also serves as a central place to satisfy the need for insert, update, or delete of cost information records in OPM from any source.

Following are some of the important characteristics that these APIs have:

Code Re-Use

You can maximize code reuse from all application development tools, including PL/SQL, Oracle Forms, and Oracle Reports.

Ease of Integration

You can integrate APIs into other applications and enabling technology, such as Oracle Workflow Server, Oracle Internet Commerce & Oracle WebSystem, and Oracle EDI Gateway.

Insulation from Changes

You can encapsulate the structure of schema to prevent changing schema structures from affecting other applications.

Consistent Behavior

You can hide object logic specific to an application from other applications, and to ensure that this logic is correctly invoked by other applications and customers.

Robust Validation

You can fully validate all incoming information into Oracle Applications.

Major Features

In order to support requirements mentioned north “Basic Business Needs” section, new APIs support the following functionality on business objects - **Item Cost, Resource Cost**, **Allocation Definition**, **Lot Cost Adjustments**, and **Burden Details**:

Item Cost APIs

Insert Item Cost

This API inserts new Item Cost records in OPM. It loads Item Cost information into OPM from external systems.

Update Item Cost

This API updates Item Cost record for an existing Item. You can update Item Costs in OPM.

Delete Item Cost

This API deletes Item Cost record in OPM. You can mark the Item Cost record for purge.

Retrieve Item Cost

This API retrieves Item Cost information from OPM. You can retrieve Item Cost information from OPM for external systems or for any other purpose.

Note: This reflects cost information directly from the cost details tables. This must not be used to reconcile subledger bookings or inventory valuations.

Resource Cost APIs

Insert Resource Cost

This API insert new Resource Cost records in OPM. It loads Resource Cost information into OPM from external systems.

Update Resource Cost

This API updates Resource Cost record for an existing Resource. You can update Resource Costs in OPM.

Delete Resource Cost

This API deletes Resource Cost record in OPM. You can mark the Resource Cost record for purge.

Retrieve Resource Cost

This API retrieves Resource Cost information from OPM. You can retrieve Resource Cost information from OPM to be used by external systems or for any other purpose.

Allocation Definition APIs

Insert Allocation Definition

This API inserts new Allocation Definition records in OPM. It loads allocation definition information into OPM from external systems.

Update Allocation Definition

This API updates an existing Allocation Definition record. You can update Allocation Definitions in OPM.

Delete Allocation Definition

This API deletes Allocation Definition record in OPM. You can mark the Allocation Definition record for purge.

Burden Details APIs

Insert Burden Details

This API inserts new Burden Details in OPM. It loads Burden Details information into OPM from external systems.

Update Burden Details

This API updates Burden Details for an existing item and resource. You can update Burden Details in OPM.

Delete Burden Details

This API deletes Burden Details record in OPM. You can mark the Burden Details record for purge.

Retrieve Burden Details

This API retrieves Burden Details information from OPM. You can retrieve Burden Details information from OPM to be used by external systems or for any other purpose.

Lot Cost Adjustment APIs

Insert Lot Cost Adjustment

This API inserts new Lot Cost Adjustment records in OPM. It loads Lot Cost Adjustment information into OPM from external systems.

Update Lot Cost Adjustment

This API updates Lot Cost Adjustment records for an existing item.

Delete Lot Cost Adjustment

This API deletes Lot Cost Adjustment records in OPM.

Retrieve Lot Cost Adjustment

This API retrieves Lot Cost Adjustment information from OPM. You can retrieve Lot Cost Adjustment information from OPM to be used by external systems.

Cost Management API Features

- Creating Updating and Deleting Information
- Proper Encapsulation
- Synchronous Processing Following the Business Hierarchy
- Detailed and Translatable Error Messages

Cost Management API Support Policy

Cost Management APIs are supported by Oracle. This means:

- Oracle provides objects and libraries needed to use to the APIs and the documentation for their use.
- Oracle ensures that the APIs function as designed.
- Oracle does not support customer generated programs that use the APIs.

Oracle Applications Packages Supplied

Cost Management APIs make use of the following standard Oracle Applications packages:

- FND_API - the standard Oracle Applications API version checking function. This is used by the stored procedure to check valid API version number and also contains constant variables such as TRUE and FALSE.
- FND_MESSAGE - the standard Oracle Applications messaging function. This is used by the stored procedure to report status and error handling.
- FND_PUB_MSG - the standard Oracle Applications message retrieval function, used to interrogate the procedure messages.

These packages are installed as part of the current release. Please refer to the *Oracle Applications Coding Standards* document for additional details.

Cost Management API Bill of Materials

The following is a list of packages and files that are delivered with OPM Cost Management APIs. These must be on your system for your interface to compile and link properly.

Package Name	File Names	Description
GMF_ItemCost_PUB	GMFPCSTS.pls GMFPCSTB.pls	Public Cost Management package that the user defined function calls. The business API can be used for inserting, updating, deleting, or retrieving an item cost.
GMF_ResourceCost_PUB	GMFPRESS.pls GMFPCSTB.pls	Public Cost Management package that the user defined function calls. The business API can be used for inserting, updating, deleting, or retrieving a resource cost.
GMF_AllocationDefinition_PUB	GMFPALCS.pls GMFPCSTB.pls	Public Cost Management package that the user defined function calls. The business API can be used for inserting, updating, or deleting an allocation definition.
GMF_BurdenDetails_PUB	GMFPBRDS.pls GMFPBRDB.pls	Public Cost Management package that the user defined function calls. The business API can be used for inserting, updating, deleting, or retrieving a burden detail.

Package Name	File Names	Description
GMF_LotCostAdjustment_PUB	GMFPLCAS.pls GMFPLCAB.pls	Public Cost Management package that the user-defined function calls. The business API can be used for inserting, updating, deleting, or retrieving lot cost adjustments.

2

Cost Management API Usage

The Cost Management APIs are currently written in PL/SQL. To make use of these APIs, you must code your interface or wrapper. Your program is also responsible for connecting to a database before calling an API function. You may also choose to write log files before calling and after returning from an API function. Each function returns an error code in the parameter x_return_status which indicates whether the API was successful or failed. The values are as follows:

S - Successful

E - Expected error

U - Unexpected error

Calling the API Interface Code

The following are snippets from a sample wrapper (referred to as wrapper from here on) and are used to test the API code. Wrappers are written in PL/SQL Package. Wrappers can be written for each API and call the APIs directly from these wrappers. The source of data for the wrapper comes from an ASCII flat file. You can write a similar type of wrapper to call the API code.

These wrappers have following parameters :

p_dir IN VARCHAR2 - Working directory for input and output files.

p_input_file IN VARCHAR2 - Name of input file, that is, ASCII flat file with data

p_output_file IN VARCHAR2 - Name of output file

p_delimiter IN VARCHAR2 - Delimiter character

The ASCII flat file must be 'character delimited' (typically, but not necessarily, with a comma).

Calling the API Code - Example

This section details how to call the API code within the wrapper. The purpose of this is to explain how to call a standard OPM Cost Management APIs. Here Item Cost API is taken as an example. The wrapper package consists of a PL/SQL function named 'Create_Item_Cost' which returns VARCHAR2 indicating return status.

It is not critical how the data values get into your customized interface. What is dictated is the order in which those same data values get passed into the standard API calls.

Typically, wrappers goes through following steps for each item:

1. Load the data from the flat file into PL/SQL table or record.
2. Call the Item Cost API for each item.
3. Check the return status.
4. Retrieve the messages from the message stack and write them into a log file.

Wrapper logic

1. Lines 66-69, declares PL/SQL variables to hold the data coming from flat file. These variables gets passed to the public API, so that data type corresponds with the data types specified in the API specification.
2. Lines 125-132, get the first line from the flat file and extract the first value. In our case it is a type of the record. Following are the possible values:

   10 : Item Cost Header record

   20 : This Level record for the Item

   30 : Lower Level record for the Item.
3. Lines 143-261, populate the item details into PL/SQL tables based on the type of record.
4. Lines 263-281, error handling. If error is raised while populating header record, then skip all the detail records. If detail record, then skip only that detail record and continue populating other detail records.
5. Lines 283-335, get the next line from the file and check type of record. Following scenarios may occur :

   - If the record type is of detail, then continue populating the record.

- If the record type is header record, then call the public API for the previous Item Cost Detail already read.
- If the fetch fails (end of file is reached - No_Data_Found error), then call the public API for the previous Item Cost Detail already read.

6. Lines 343-373, read all the messages from the message stack and write it into a log file.
7. Lines 376-383, check the out parameter l_costcmpnt_ids PL/SQL table, if any records are inserted then get the information for what detail records have been inserted.
8. Lines 385-389, check whether to continue the loop or not. If yes, then start over again from step 2.

API Hints

For performance improvement, NOCOPY hints have been added to the OUT parameters of the APIs. When an API has the same type of parameter defined as IN and OUT, you must pass in different variables. In addition, you must check the return status of the API (generally returned through x_return_status parameter) before looking at other OUT variables returned by the API. If the return status is not Success, then you must not use any of the OUT parameters passed back from the API.

For example, the Get_Resource_Cost API contains p_resource_cost_rec and x_resource_cost_rec:

```
PROCEDURE Get_Resource_Cost
( p_api_version  IN  NUMBER    ,
 p_init_msg_list  IN  VARCHAR2 := FND_API.G_FALSE  ,
 x_return_status  OUT NOCOPY VARCHAR2    ,
 x_msg_count  OUT NOCOPY NUMBER    ,
 x_msg_data  OUT NOCOPY VARCHAR2    ,
 p_resource_cost_rec IN  Resource_Cost_Rec_Type  ,
 x_resource_cost_rec OUT NOCOPY Resource_Cost_Rec_Type
);
```

Therefore, the call can be set up to read:

```
GMF_RESOURCECOST_PUB.get_resource_cost(
        (p_api_version => l_api_version,
        p_init_msg_list => l_init_msg_list,
        x_return_status => l_return_status,
        x_msg_count => l_msg_count,
        x_msg_data => l_msg_data,
        p_resource_cost_rec => l_resource_cost_rec,
        x_resource_cost_rec => l_resource_cost_rec );
```

In this example, p_resource_cost_rec and x_resource_cost_rec both have the variable l_resource_cost_rec. This gives an incorrect result because both the parameters cannot have the same variable.

You must set the call up so p_resource_cost_rec and x_resource_cost_rec have different variables:

```
GMF_RESOURCECOST_PUB.get_resource_cost(
        (p_api_version => l_api_version,
        p_init_msg_list => l_init_msg_list,
        x_return_status => l_return_status,
        x_msg_count => l_msg_count,
        x_msg_data => l_msg_data,
```

```
        p_resource_cost_rec => l_resource_cost_rec_in,
        x_resource_cost_rec => l_resource_cost_rec_out );
```

Item Cost Example

```
1
--+============================================================================+
2    --| FUNCTION NAME
|
3    --|    Create_Item_Cost
|
4    --|
|
5    --| TYPE
|
6    --|    Public
|
7    --|
|
8    --| USAGE
|
9    --|    Create item Cost
|
10   --|
|
11   --| DESCRIPTION
|
12   --|    This is a PL/SQL wrapper function to call the
|
13   --|    Create_Item_Cost API.
|
14   --|    It reads item data from a flat file and outputs any error
|
15   --|    messages to a second flat file. It also generates a log file
|
16   --|   called gmf_api_cric_wrapper<session_id>.log in the p_dir directory.
|
17   --|
|
18   --| PARAMETERS
|
19   --|    p_dir              IN VARCHAR2         - Working directory for input
|
20   --|                                             and output files.
```

```
|
21   --|    p_input_file       IN VARCHAR2         - Name of input file
|
22   --|    p_output_file      IN VARCHAR2         - Name of output file
|
23   --|    p_delimiter        IN VARCHAR2         - Delimiter character
|
24   --|
|
25   --| RETURNS
|
26   --|    VARCHAR2 - 'S' All records processed successfully
|
27   --|               'E' 1 or more records errored
|
28   --|               'U' 1 or more record unexpected error
|
29   --|
|
30   --| HISTORY
|
31   --|
|
32
--+=============================================================================+
33   --Api end of comments
34
35   FUNCTION Create_Item_Cost
36   ( p_dir          IN VARCHAR2
37   , p_input_file   IN VARCHAR2
38   , p_output_file  IN VARCHAR2
39   , p_delimiter    IN VARCHAR2
40   )
41   RETURN VARCHAR2
42   IS
43
44   /* Local variables */
45
46   l_status             VARCHAR2(11);
47   l_return_status      VARCHAR2(11) :=FND_API.G_RET_STS_SUCCESS;
48   l_count              NUMBER(10)  ;
49   l_record_count       NUMBER(10)  :=0;
50   l_loop_cnt           NUMBER(10)  :=0;
51   l_dummy_cnt          NUMBER(10)  :=0;
52   l_data               VARCHAR2(1000);
```

```
53
54   l_p_dir              VARCHAR2(150);
55   l_output_file        VARCHAR2(120);
56   l_outfile_handle     UTL_FILE.FILE_TYPE;
57   l_input_file         VARCHAR2(120);
58   l_infile_handle      UTL_FILE.FILE_TYPE;
59   l_line               VARCHAR2(1000);
60   l_delimiter          VARCHAR(11);
61   l_log_dir            VARCHAR2(150);
62   l_log_name           VARCHAR2(120)  :='gmf_api_cric_wrapper';
63   l_log_handle         UTL_FILE.FILE_TYPE;
64   l_global_file        VARCHAR2(120);
65
66   l_header_rec         GMF_ItemCost_PUB.Header_Rec_Type;
67   l_this_lvl_tbl       GMF_ItemCost_PUB.This_Level_Dtl_Tbl_Type;
68   l_lower_lvl_tbl      GMF_ItemCost_PUB.Lower_Level_Dtl_Tbl_Type;
69   l_costcmpnt_ids      GMF_ItemCost_PUB.costcmpnt_ids_tbl_type;
70   l_idx     NUMBER(10);
71   l_idx1     NUMBER(10);
72   l_type     NUMBER(10);
73   l_continue           VARCHAR2(1) := 'Y' ;
74   l_skip_details       VARCHAR2(1) := 'N' ;
75
76   l_session_id         VARCHAR2(110);
77
78   BEGIN
79
80   /*  Enable The Buffer  */
81   DBMS_OUTPUT.ENABLE(1000000);
82
83   l_p_dir              :=p_dir;
84   l_input_file         :=p_input_file;
85   l_output_file        :=p_output_file;
86   l_delimiter          :=p_delimiter;
87   l_global_file        :=l_input_file;
88
89   /*  Obtain The SessionId To Append To wrapper File Name. */
90
91   l_session_id := USERENV('sessionid');
92
93   l_log_name  := CONCAT(l_log_name,l_session_id);
94   l_log_name  := CONCAT(l_log_name,'.log');
95
96   /*  Directory is now the same same as for the out file */
97   l_log_dir   := p_dir;
```

```
98
99
100  /*  Open The Wrapper File For Output And The Input File for Input. */
101
102  l_log_handle      :=UTL_FILE.FOPEN(l_log_dir, l_log_name, 'w');
103  l_infile_handle   :=UTL_FILE.FOPEN(l_p_dir, l_input_file, 'r');
104
105  /*  Loop thru flat file and call Item Cost API */
106  DBMS_OUTPUT.PUT_LINE('Process Started at ' || to_char(SYSDATE,'DD-MON-YY
HH24:MI:SS'));
107  DBMS_OUTPUT.PUT_LINE('Input Directory  ' || l_p_dir );
108  DBMS_OUTPUT.PUT_LINE('Input File       ' || l_input_file );
109  DBMS_OUTPUT.PUT_LINE('Delimiter        ' || l_delimiter );
110  DBMS_OUTPUT.PUT_LINE('Output File      ' || l_output_file );
111
112
113  DBMS_OUTPUT.PUT_LINE('Start Processing');
114  UTL_FILE.PUT_LINE(l_log_handle, 'Process Started at ' || to_
char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
115
116  UTL_FILE.NEW_LINE(l_log_handle);
117  UTL_FILE.PUT_LINE(l_log_handle, 'Input Directory  ' || l_p_dir );
118  UTL_FILE.PUT_LINE(l_log_handle, 'Input File       ' || l_input_file );
119  UTL_FILE.PUT_LINE(l_log_handle, 'Record Type      ' || l_delimiter );
120  UTL_FILE.PUT_LINE(l_log_handle, 'Output File      ' || l_output_file );
121
122  l_outfile_handle  :=UTL_FILE.FOPEN(l_p_dir, l_output_file, 'w');
123
124   /* Get the first record from the file */
125   BEGIN
126     UTL_FILE.GET_LINE(l_infile_handle, l_line);
127     l_record_count    :=l_record_count+1;
128     l_type   := Get_Field(l_line,l_delimiter,1) ;  -- 10 : header rec, 20 :
this level, 30 : lower level
129   EXCEPTION
130     WHEN NO_DATA_FOUND THEN
131       raise;
132   END;
133
134   /* Populate appropriate pl/sql record or table based on the type of record
*/
135   LOOP
136    BEGIN
137
138      UTL_FILE.PUT_LINE(l_log_handle, 'Reading Record...' || l_record_count
```

```
);
139      UTL_FILE.PUT_LINE(l_log_handle, 'Type   = ' || l_type) ;
140
141      /* Header record */
142
143      IF l_type = 10 THEN
144        -- empty the pl/sql tables to remove details of the previous item
145        -- and initialize the index
146        l_this_lvl_tbl.delete ;
147        l_lower_lvl_tbl.delete ;
148        l_costcmpnt_ids.delete;
149        l_skip_details := 'N' ;
150        l_idx  := 0 ;
151        l_idx1 := 0 ;
152
153        l_header_rec.calendar_code    := Get_Field(l_line,l_delimiter,2) ;
154        l_header_rec.period_code      := Get_Field(l_line,l_delimiter,3) ;
155        l_header_rec.cost_mthd_code   := Get_Field(l_line,l_delimiter,4) ;
156        l_header_rec.whse_code        := Get_Field(l_line,l_delimiter,5) ;
157        l_header_rec.item_id          := Get_Field(l_line,l_delimiter,6) ;
158        l_header_rec.item_no          := Get_Field(l_line,l_delimiter,7) ;
159        l_header_rec.user_name        := Get_Field(l_line,l_delimiter,8) ;
160
161        UTL_FILE.PUT_LINE(l_log_handle, 'Type            = ' || l_type) ;
162        UTL_FILE.PUT_LINE(l_log_handle, 'calendar_code   = ' || l_header_
rec.calendar_code) ;
163        UTL_FILE.PUT_LINE(l_log_handle, 'period_code     = ' || l_header_
rec.period_code) ;
164        UTL_FILE.PUT_LINE(l_log_handle, 'cost_mthd_code  = ' || l_header_
rec.cost_mthd_code) ;
165        UTL_FILE.PUT_LINE(l_log_handle, 'whse_code       = ' || l_header_
rec.whse_code) ;
166        UTL_FILE.PUT_LINE(l_log_handle, 'item_id         = ' || l_header_
rec.item_id) ;
167        UTL_FILE.PUT_LINE(l_log_handle, 'item_no         = ' || l_header_
rec.item_no) ;
168        UTL_FILE.PUT_LINE(l_log_handle, 'user_name       = ' || l_header_
rec.user_name) ;
169
170      /* This Level Detail record. Skip details in case errors in loading
header record */
171      ELSIF l_type = 20 AND l_skip_details = 'Y' THEN
172        UTL_FILE.PUT_LINE(l_log_handle, 'Error : Skipping this record...');
173      ELSIF l_type = 20 AND l_skip_details = 'N' THEN
174        l_idx := l_idx + 1 ;
```

```
175        l_this_lvl_tbl(l_idx).cmpntcost_id         := Get_Field(l_line,l_
delimiter,2) ;
176        l_this_lvl_tbl(l_idx).cost_cmpntcls_id     := Get_Field(l_line,l_
delimiter,3) ;
177        l_this_lvl_tbl(l_idx).cost_cmpntcls_code   := Get_Field(l_line,l_
delimiter,4) ;
178        l_this_lvl_tbl(l_idx).cost_analysis_code   := Get_Field(l_line,l_
delimiter,5) ;
179        l_this_lvl_tbl(l_idx).cmpnt_cost           := Get_Field(l_line,l_
delimiter,6) ;
180        l_this_lvl_tbl(l_idx).burden_ind           := Get_Field(l_line,l_
delimiter,7) ;
181        l_this_lvl_tbl(l_idx).total_qty            := Get_Field(l_line,l_
delimiter,8) ;
182        l_this_lvl_tbl(l_idx).costcalc_orig        := Get_Field(l_line,l_
delimiter,9) ;
183        l_this_lvl_tbl(l_idx).rmcalc_type          := Get_Field(l_line,l_
delimiter,10) ;
184        l_this_lvl_tbl(l_idx).delete_mark          := Get_Field(l_line,l_
delimiter,11) ;
185        l_this_lvl_tbl(l_idx).attribute1           := Get_Field(l_line,l_
delimiter,12) ;
186        l_this_lvl_tbl(l_idx).attribute2           := Get_Field(l_line,l_
delimiter,13) ;
187        l_this_lvl_tbl(l_idx).attribute3           := Get_Field(l_line,l_
delimiter,14) ;
188        l_this_lvl_tbl(l_idx).attribute4           := Get_Field(l_line,l_
delimiter,15) ;
189        l_this_lvl_tbl(l_idx).attribute5           := Get_Field(l_line,l_
delimiter,16) ;
190        l_this_lvl_tbl(l_idx).attribute6           := Get_Field(l_line,l_
delimiter,17) ;
191        l_this_lvl_tbl(l_idx).attribute7           := Get_Field(l_line,l_
delimiter,18) ;
192        l_this_lvl_tbl(l_idx).attribute8           := Get_Field(l_line,l_
delimiter,19) ;
193        l_this_lvl_tbl(l_idx).attribute9           := Get_Field(l_line,l_
delimiter,20) ;
194        l_this_lvl_tbl(l_idx).attribute10          := Get_Field(l_line,l_
delimiter,21) ;
195        l_this_lvl_tbl(l_idx).attribute11          := Get_Field(l_line,l_
delimiter,22) ;
196        l_this_lvl_tbl(l_idx).attribute12          := Get_Field(l_line,l_
delimiter,23) ;
197        l_this_lvl_tbl(l_idx).attribute13          := Get_Field(l_line,l_
```

```
delimiter,24) ;
198        l_this_lvl_tbl(l_idx).attribute14          := Get_Field(l_line,l_
delimiter,25) ;
199        l_this_lvl_tbl(l_idx).attribute15          := Get_Field(l_line,l_
delimiter,26) ;
200        l_this_lvl_tbl(l_idx).attribute16          := Get_Field(l_line,l_
delimiter,27) ;
201        l_this_lvl_tbl(l_idx).attribute17          := Get_Field(l_line,l_
delimiter,28) ;
202        l_this_lvl_tbl(l_idx).attribute18          := Get_Field(l_line,l_
delimiter,29) ;
203        l_this_lvl_tbl(l_idx).attribute19          := Get_Field(l_line,l_
delimiter,30) ;
204        l_this_lvl_tbl(l_idx).attribute20          := Get_Field(l_line,l_
delimiter,31) ;
205        l_this_lvl_tbl(l_idx).attribute21          := Get_Field(l_line,l_
delimiter,32) ;
206        l_this_lvl_tbl(l_idx).attribute22          := Get_Field(l_line,l_
delimiter,33) ;
207        l_this_lvl_tbl(l_idx).attribute23          := Get_Field(l_line,l_
delimiter,34) ;
208        l_this_lvl_tbl(l_idx).attribute24          := Get_Field(l_line,l_
delimiter,35) ;
209        l_this_lvl_tbl(l_idx).attribute25          := Get_Field(l_line,l_
delimiter,36) ;
210        l_this_lvl_tbl(l_idx).attribute26          := Get_Field(l_line,l_
delimiter,37) ;
211        l_this_lvl_tbl(l_idx).attribute27          := Get_Field(l_line,l_
delimiter,38) ;
212        l_this_lvl_tbl(l_idx).attribute28          := Get_Field(l_line,l_
delimiter,39) ;
213        l_this_lvl_tbl(l_idx).attribute29          := Get_Field(l_line,l_
delimiter,40) ;
214        l_this_lvl_tbl(l_idx).attribute30          := Get_Field(l_line,l_
delimiter,41) ;
215        l_this_lvl_tbl(l_idx).attribute_category   := Get_Field(l_line,l_
delimiter,42) ;
216
217        UTL_FILE.PUT_LINE(l_log_handle,'cmpntcost_id('||l_idx||')       = '
||
218          l_this_lvl_tbl(l_idx).cmpntcost_id) ;
219        UTL_FILE.PUT_LINE(l_log_handle,'cost_cmpntcls_id('||l_idx||')   = '
||
220          l_this_lvl_tbl(l_idx).cost_cmpntcls_id) ;
221        UTL_FILE.PUT_LINE(l_log_handle,'cost_cmpntcls_code('||l_idx||') = '||
```

```
222  l_this_lvl_tbl(l_idx).cost_cmpntcls_code) ;
223        UTL_FILE.PUT_LINE(l_log_handle,'cost_analysis_code('||l_idx||') = '||
224  l_this_lvl_tbl(l_idx).cost_analysis_code) ;
225        UTL_FILE.PUT_LINE(l_log_handle,'cmpnt_cost('||l_idx||')         = '
||
226          l_this_lvl_tbl(l_idx).cmpnt_cost) ;
227        UTL_FILE.PUT_LINE(l_log_handle,'burden_ind('||l_idx||')         = '
||
228          l_this_lvl_tbl(l_idx).burden_ind) ;
229        UTL_FILE.PUT_LINE(l_log_handle,'total_qty('||l_idx||')          = '
||
230          l_this_lvl_tbl(l_idx).total_qty) ;
231        UTL_FILE.PUT_LINE(l_log_handle,'costcalc_orig('||l_idx||')      = '
||
232          l_this_lvl_tbl(l_idx).costcalc_orig) ;
233        UTL_FILE.PUT_LINE(l_log_handle,'rmcalc_type('||l_idx||')        = '
||
234          l_this_lvl_tbl(l_idx).rmcalc_type) ;
235        UTL_FILE.PUT_LINE(l_log_handle,'delete_mark('||l_idx||')        = '
||
236          l_this_lvl_tbl(l_idx).delete_mark) ;
237
238      /* Lower Level Detail record. Skip details in case errors in loading
header record */
239      ELSIF l_type = 30 AND l_skip_details = 'Y' THEN
240        UTL_FILE.PUT_LINE(l_log_handle, 'Error : Skipping this record...');
241      ELSIF l_type = 30 AND l_skip_details = 'N'  THEN
242        l_idx1 := l_idx1 + 1 ;
243        l_type                                      := Get_Field(l_line,l_
delimiter,1) ;
244        l_lower_lvl_tbl(l_idx1).cmpntcost_id        := Get_Field(l_line,l_
delimiter,2) ;
245        l_lower_lvl_tbl(l_idx1).cost_cmpntcls_id    := Get_Field(l_line,l_
delimiter,3) ;
246        l_lower_lvl_tbl(l_idx1).cost_cmpntcls_code  := Get_Field(l_line,l_
delimiter,4) ;
247        l_lower_lvl_tbl(l_idx1).cost_analysis_code  := Get_Field(l_line,l_
delimiter,5) ;
248        l_lower_lvl_tbl(l_idx1).cmpnt_cost          := Get_Field(l_line,l_
delimiter,6) ;
249
250        UTL_FILE.PUT_LINE(l_log_handle,'ll cmpntcost_id('||l_idx1||')      =
' ||
251          l_lower_lvl_tbl(l_idx1).cmpntcost_id) ;
252        UTL_FILE.PUT_LINE(l_log_handle,'ll cost_cmpntcls_id('||l_idx1||')  =
```

```
' ||
253          l_lower_lvl_tbl(l_idx1).cost_cmpntcls_id)      ;
254       UTL_FILE.PUT_LINE(l_log_handle,'ll cost_cmpntcls_code('||l_idx1||') =
'||
255          l_lower_lvl_tbl(l_idx1).cost_cmpntcls_code) ;
256       UTL_FILE.PUT_LINE(l_log_handle,'ll cost_analysis_code('||l_idx1||') =
'||
257  l_lower_lvl_tbl(l_idx1).cost_analysis_code) ;
258       UTL_FILE.PUT_LINE(l_log_handle,'ll cmpnt_cost('||l_idx1||')        =
' ||
259          l_lower_lvl_tbl(l_idx1).cmpnt_cost) ;
260
261      END IF ;
262
263    EXCEPTION
264     WHEN OTHERS THEN
265      /* in case of any errors log the error and continue with the next
record
266       * in case of header, skip all the detail record
267       * in case details, skip that particular record */
268
269     UTL_FILE.PUT_LINE(l_outfile_handle, 'Error : ' || to_char(SQLCODE) || '
' || SQLERRM);
270      UTL_FILE.PUT_LINE(l_log_handle, 'Error : '     || to_char(SQLCODE) || '
' || SQLERRM);
271
272      IF l_type = 10 THEN
273        l_skip_details := 'Y' ;
274        UTL_FILE.PUT_LINE(l_log_handle, 'Error : Skip detail records.');
275      ELSIF l_type = 20 THEN
276        l_this_lvl_tbl.delete(l_idx);
277      ELSIF l_type = 30 THEN
278        l_lower_lvl_tbl.delete(l_idx1);
279      END IF ;
280
281    END ;
282
283    BEGIN
284
285        UTL_FILE.GET_LINE(l_infile_handle, l_line);
286        l_record_count    :=l_record_count+1;
287        UTL_FILE.NEW_LINE(l_log_handle);
288        l_type   := Get_Field(l_line,l_delimiter,1) ;  -- 10 : header rec, 20
: this level, 30 : lower level
289
```

```
290    EXCEPTION
291     /* End of file */
292     WHEN NO_DATA_FOUND THEN
293      IF l_skip_details = 'N' THEN
294
295        GMF_ItemCost_PUB.Create_Item_Cost
296        ( p_api_version         => 1.0
297        , p_init_msg_list       => FND_API.G_TRUE
298        , p_commit              => FND_API.G_TRUE
299
300        , x_return_status       => l_status
301        , x_msg_count           => l_count
302        , x_msg_data            => l_data
303
304        , p_header_rec          => l_header_rec
305        , p_this_level_dtl_tbl  => l_this_lvl_tbl
306        , p_lower_level_dtl_Tbl => l_lower_lvl_tbl
307
308        , x_costcmpnt_ids       => l_costcmpnt_ids
309        );
310        l_continue := 'N' ;
311        goto GET_MSG_STACK ;
312      END IF ;
313    END;
314
315    IF (l_type = 10 AND l_record_count != 1 AND l_skip_details = 'N') THEN
316
317      DBMS_OUTPUT.PUT_LINE('Calling Create_Item_Cost API...');
318
319      GMF_ItemCost_PUB.Create_Item_Cost
320      ( p_api_version         => 1.0
321      , p_init_msg_list       => FND_API.G_TRUE
322      , p_commit              => FND_API.G_TRUE
323
324      , x_return_status       => l_status
325      , x_msg_count           => l_count
326      , x_msg_data            => l_data
327
328      , p_header_rec          => l_header_rec
329      , p_this_level_dtl_tbl  => l_this_lvl_tbl
330      , p_lower_level_dtl_Tbl => l_lower_lvl_tbl
331
332      , x_costcmpnt_ids       => l_costcmpnt_ids
333      );
334      DBMS_OUTPUT.PUT_LINE('after API call. status := ' || l_status ||' cnt
```

```
:= ' || l_count );
335    END IF;
336
337    <<GET_MSG_STACK>>
338      NULL;
339
340    /*  Check if any messages generated. If so then decode and */
341    /*  output to error message flat file */
342
343    IF l_count > 0 THEN
344
345      l_loop_cnt  :=1;
346      LOOP
347        FND_MSG_PUB.Get(
348          p_msg_index     => l_loop_cnt,
349          p_data          => l_data,
350          p_encoded       => FND_API.G_FALSE,
351          p_msg_index_out => l_dummy_cnt);
352
353        --DBMS_OUTPUT.PUT_LINE(l_data );
354        UTL_FILE.PUT_LINE(l_log_handle, l_data);
355
356        /*  Update error status */
357        IF (l_status = 'U') THEN
358          l_return_status  :=l_status;
359        ELSIF (l_status = 'E' and l_return_status <> 'U') THEN
360          l_return_status  :=l_status;
361        ELSE
362          l_return_status  :=l_status;
363        END IF;
364
365        l_loop_cnt  := l_loop_cnt + 1;
366        IF l_loop_cnt > l_count THEN
367          EXIT;
368        END IF;
369
370      END LOOP; -- msg stack loop
371      l_count := 0 ;
372
373    END IF;-- if count of msg stack > 0
374
375    DBMS_OUTPUT.PUT_LINE('# of CostIds inserted : ' || l_costcmpnt_
ids.count);
376    FOR i in 1..l_costcmpnt_ids.count
377    LOOP
```

```
378      UTL_FILE.PUT_LINE(l_log_handle,
379        'CmpntClsId : '     || l_costcmpnt_ids(i).cost_cmpntcls_id ||
380        ' Analysis Code : ' || l_costcmpnt_ids(i).cost_analysis_code ||
381        ' Cost Level : '    || l_costcmpnt_ids(i).cost_level ||
382        ' CostId : '        || l_costcmpnt_ids(i).cmpntcost_id);
383    END LOOP ;
384
385    IF l_continue = 'N' THEN
386      EXIT ;
387    END IF ;
388
389   END LOOP;
390
391    UTL_FILE.NEW_LINE(l_log_handle);
392    UTL_FILE.PUT_LINE(l_log_handle, 'Process Completed at ' || to_
char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
393    UTL_FILE.FCLOSE_ALL;
394
395    RETURN l_return_status;
396
397  EXCEPTION
398  WHEN UTL_FILE.INVALID_OPERATION THEN
399     /*  DBMS_OUTPUT.PUT_LINE('Invalid Operation For '|| l_global_file); */
400     UTL_FILE.FCLOSE_ALL;
401     RETURN l_return_status;
402
403  WHEN UTL_FILE.INVALID_PATH THEN
404     /*  DBMS_OUTPUT.PUT_LINE('Invalid Path For      '|| l_global_file); */
405     UTL_FILE.FCLOSE_ALL;
406     RETURN l_return_status;
407
408  WHEN UTL_FILE.INVALID_MODE THEN
409     /*  DBMS_OUTPUT.PUT_LINE('Invalid Mode For      '|| l_global_file); */
410     UTL_FILE.FCLOSE_ALL;
411     RETURN l_return_status;
412
413  WHEN UTL_FILE.INVALID_FILEHANDLE THEN
414     /* DBMS_OUTPUT.PUT_LINE('Invalid File Handle   '|| l_global_file); */
415     UTL_FILE.FCLOSE_ALL;
416     RETURN l_return_status;
417
418  WHEN UTL_FILE.WRITE_ERROR THEN
419     /*  DBMS_OUTPUT.PUT_LINE('Invalid Write Error   '|| l_global_file); */
420     UTL_FILE.FCLOSE_ALL;
421     RETURN l_return_status;
```

```
422
423  WHEN UTL_FILE.READ_ERROR THEN
424     /*  DBMS_OUTPUT.PUT_LINE('Invalid Read  Error   '|| l_global_file); */
425     UTL_FILE.FCLOSE_ALL;
426     RETURN l_return_status;
427
428  WHEN UTL_FILE.INTERNAL_ERROR THEN
429     /*  DBMS_OUTPUT.PUT_LINE('Internal Error'); */
430     UTL_FILE.FCLOSE_ALL;
431     RETURN l_return_status;
432
433  WHEN OTHERS THEN
434     /*  DBMS_OUTPUT.PUT_LINE('Other Error'); */
435    UTL_FILE.PUT_LINE(l_outfile_handle, 'Error : ' || to_char(SQLCODE) || ' '
|| SQLERRM);
436    UTL_FILE.PUT_LINE(l_log_handle, 'Error : ' || to_char(SQLCODE) || ' ' ||
SQLERRM);
437    UTL_FILE.PUT_LINE(l_log_handle, 'Process Completed at ' || to_
char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
438    UTL_FILE.FCLOSE_ALL;
439    l_return_status := 'U' ;
440    RETURN l_return_status;
441
442  END Create_Item_Cost;
443
444
445
--+============================================================================+
446  --| FUNCTION NAME
|
447  --|    Get_Field
|
448  --|
|
449  --| TYPE
|
450  --|    Public
|
451  --|
|
452  --| USAGE
|
453  --|    Get value of field n from a delimited line of ASCII data
|
454  --|
```

```
|
455  --| DESCRIPTION
|
456  --|    This utility function will return the value of a field from
|
457  --|    a delimited line of ASCII text
|
458  --|
|
459  --| PARAMETERS
|
460  --|    p_line             IN VARCHAR2         - line of data
|
461  --|    p_delimiter        IN VARCHAR2         - Delimiter character
|
462  --|    p_field_no         IN NUMBER           - Field occurance to be
|
463  --|                                             returned
|
464  --|
|
465  --| RETURNS
|
466  --|    VARCHAR2                               - Value of field
|
467  --|
|
468  --| HISTORY
|
469  --|
|
470
--+============================================================================+
471  -- Api end of comments
472
473  FUNCTION Get_Field
474  ( p_line         IN VARCHAR2
475  , p_delimiter    IN VARCHAR2
476  , p_field_no     IN NUMBER
477  )
478  RETURN VARCHAR2
479  IS
480
481  /*  Local variables */
482  l_start         NUMBER  :=0;
```

```
l_end           NUMBER  :=0;

BEGIN

/* Determine start position */
IF p_field_no = 1
THEN
  l_start       :=0;
ELSE
  l_start       :=INSTR(p_line,p_delimiter,1,(p_field_no - 1));
  IF l_start    = 0
  THEN
    RETURN NULL;
  END IF;
END IF;

/*  Determine end position */
l_end           :=INSTR(p_line,p_delimiter,1,p_field_no);
IF l_end        = 0
THEN
  l_end         := LENGTH(p_line) + 1;
END IF;

/*  Extract the field data */
IF (l_end - l_start) = 1
THEN
  RETURN NULL;
ELSE
  RETURN SUBSTR(p_line,(l_start + 1),((l_end - l_start) - 1));
END IF;

EXCEPTION
  WHEN OTHERS
  THEN
    RETURN NULL;

END Get_Field;


--+============================================================================+
--| FUNCTION NAME
|
--|    Get_Substring
|
--|
```

```
|
525  --| TYPE
|
526  --|    Public
|
527  --|
|
528  --| USAGE
|
529  --|    Get value of Sub-string from formatted ASCII data file record
|
530  --|
|
531  --| DESCRIPTION
|
532  --|    This utility function will return the value of a passed sub-string
|
533  --|    of a formatted ASCII data file record
|
534  --|
|
535  --| PARAMETERS
|
536  --|    p_substring        IN VARCHAR2         - substring data
|
537  --|
|
538  --| RETURNS
|
539  --|    VARCHAR2                               - Value of field
|
540  --|
|
541  --| HISTORY
|
542  --|
|
543
--+============================================================================+
544  -- Api end of comments
545
546  FUNCTION Get_Substring
547  ( p_substring    IN VARCHAR2
548  )
549  RETURN VARCHAR2
```

```
IS

/*  Local variables */
l_string_value   VARCHAR2(200)  :=' ';

BEGIN

/*  Determine start position */
l_string_value  :=NVL(RTRIM(LTRIM(p_substring)),' ');

RETURN l_string_value;
EXCEPTION
  WHEN OTHERS
  THEN
    RETURN ' ';

END Get_Substring;
```

3

Technical Overview

Item Cost

Each function on a business object, Item Cost, is associated with a Public API, through which Item Cost details can be created, updated, deleted, and retrieved from OPM.

The Public API performs all validations on input data supplied to prevent the flow of invalid data into OPM. If there are validation errors, then the API stops processing and returns an error status without finishing the remaining detail records. During the insert (after the validations to all detail records) if the insert fails, then all the detail records for the item in process are not inserted. After finishing validations on input data, the public API performs the required function by calling necessary routines.

Structure for Item Cost Public APIs

According to API standards, the following are the names of files, packages, and procedures for Public APIs:

Object Type	Name
Package Specification File	GMFPCSTS.pls
Package Body File	GMFPCSTB.pls
Package	GMF_ItemCost_PUB
Procedure - Create Item Cost	Create_Item_Cost
Procedure - Update Item Cost	Update_Item_Cost
Procedure - Delete Item Cost	Delete_Item_Cost

Object Type	Name
Procedure - Get Item Cost	Get_Item_Cost

Resource Cost

Each function on a business object, Resource Cost, is associated with a Public API, through which Resource Cost details can be created, updated, deleted and retrieved from OPM.

The Public API performs all validations on input data supplied to prevent the flow of invalid data into OPM. If there are validation errors, then that particular row is skipped and the process continues with the next Resource Cost record. During the insert (after the validations to all detail records) if the insert fails, then the Resource Cost record in process is not inserted and process continues with the next Resource Cost record. After finishing validations on input data, the public API performs the required function by calling necessary routines.

Structure for Resource Cost Public APIs

According to API standards, the following are the names of files, packages, and procedures for Public APIs:

Object Type	Name
Package Specification File	GMFPRESS.pls
Package Body File	GMFPRESB.pls
Package	GMF_ResourceCost_PUB
Procedure - Create Resource Cost	Create_Resource_Cost
Procedure - Update Resource Cost	Update_Resource_Cost
Procedure - Delete Resource Cost	Delete_Resource_Cost
Procedure - Get Resource Cost	Get_Resource_Cost

Allocation Definition

Each function on a business object, Allocation Definition, is associated with a Public API, through which Allocation Definition details can be created, updated, deleted and retrieved from OPM.

The Public API performs all validations on input data supplied to prevent the flow of invalid data into OPM. If there are validation errors, then that row is skipped and the process continues with the next Allocation Definition record. During the insert (after the validations to all detail records) if the insert fails, then the Allocation Definition record in process is not inserted and the process continues with the next Allocation Definition record. After finishing validations on input data, the public API performs the required function by calling necessary routines.

Structure for Allocation Definition Public APIs

According to API standards, the following are the names of files, packages, and procedures for Public APIs:

Object Type	Name
Package Specification File	GMFPALCS.pls
Package Body File	GMFPALCB.pls
Package	GMF_AllocationDefinition_PUB
Procedure - Create Allocation Definition	Create_Allocation_Definition
Procedure - Update Allocation Definition	Update_Allocation_Definition
Procedure - Delete Allocation Definition	Delete_Allocation_Definition

Burden Details

Each function on a business object, Burden Details, is associated with a Public API, through which Burden Details can be created, updated, deleted, and retrieved from OPM.

The Public API performs all validations on input data supplied to prevent the flow of invalid data into OPM. If there are validation errors, then the API stops processing and returns an error status without finishing the remaining detail records. During the insert (after the validations to all detail records), if the insert fails, then all the Burden Detail records for the item in process are not inserted. After finishing validations on input data, the public API performs the required function by calling necessary routines.

Structure for Burden Details Public APIs

According to API standards, the following are the names of files, packages, and procedures for Public APIs:

Object Type	Name
Package Specification File	GMFPBRDS.pls
Package Body File	GMFPBRDB.pls
Package	GMF_BurdenDetails_PUB
Procedure - Create Burden Details	Create_Burden_Details
Procedure - Update Burden Details	Update_Burden_Details
Procedure - Delete Burden Details	Delete_Burden_Details

Lot Cost Adjustments

The Lot Cost Adjustment APIs support the insert, update, delete, and retrieval of Lot Cost Adjustments (a business object).

For each function on business object Lot Cost Adjustment, there are two types of APIs - Public API and Private API. The difference between these two is that the Public APIs go through all the validations on input parameters supplied, where as the private API does not perform any data validation on input parameters. After finishing data validation on input parameters, the public API calls the private API to perform the required function. This is required to boost the performance in cases where the API caller has already validated its data and does not need to go through those validations again.

Structure for Lot Cost Adjustment Public APIs

According to API standards, the following are the names of files, packages, and procedures for Public APIs:

Object Type	Name
Package Specification File	GMFPLCAS.pls
Package Body File	GMFPLCAB.pls
Package	GMF_LotCostAdjustment_PUB
Procedure - Create Lot Cost Adjustment	Create_LotCost_Adjustment
Procedure - Update Lot Cost Adjustment	Update_LotCost_Adjustment
Procedure - Delete Lot Cost Adjustment	Delete_LotCost_Adjustment
Procedure - Get Lot Cost Adjustment	Get_LotCost_Adjustment

Standard Parameters

API standard parameters are a collection of parameters that are common to most APIs. The following paragraphs explain the standard parameters that are used in APIs and their interpretation.

Some of the standard parameters apply to all APIs regardless of the business function they perform. For example, p_api_version and `x_return_status` is included in all APIs.

Some parameters are applicable for certain types of APIs and not applicable for other types. For example, p_commit is applicable for APIs that change the database state, and not applicable for read APIs.

Standard parameters are included in all APIs whenever applicable.

Standard IN parameters:

- p_api_version
- p_init_msg_list
- p_commit
- p_validation_level

Standard OUT parameters:

- x_return_status
- x_msg_count
- x_msg_data

Parameter	Type	IN/OUT	Required	Validation
p_api_version	varchar2	IN	Y	Validates version compatibility. The version sent by the calling function is compared to the internal version of the API and an unexpected error (U) is generated if these do not match.
p_init_msg_list	varchar2	IN	N	Used to specify whether the message list must be initialized on entry to the API. It is an optional parameter, and if not supplied, then it defaults to FND_API.G_FALSE. The API does not initialize the message list.
p_commit	varchar2	IN	N	Used to specify whether the API commits the work before returning to the calling function. If not supplied, then it defaults to FND_API.G_FALSE.

Parameter	Type	IN/OUT	Required	Validation
x_return_status	varchar2	OUT	-	Specifies whether the API was successful or failed: S - Successful E - failed due to expected error U - failed due to unexpected error
x_msg_count	number	OUT	-	Specifies number of messages added to message list.
x_msg_data	varchar2	OUT	-	Returns the messages in an encoded format. These messages are processed by the standard message functions as defined in the Business Object API Coding Standards document.

- Value-ID Conversion

IDs are used to represent primary and foreign entity keys. They are also used for internal processing of attributes. They are not meaningful to users and are usually hidden. Attributes may have values that represent them. Those values are meaningful to users and are used for display purposes. In general, APIs operate only on IDs.

Example :

An item is represented by an ID which is the NUMBER column ITEM_ID, this ID is its primary key and is used for all internal processing of the item. Besides this ID, an item is represented by a Value which is the VARCHAR2 column ITEM_NO, this value is the one displayed to users when they choose an item, thus an item can be identified by either its ID or its Value (in this case ITEM_NO).

The following set of rules are for the conversion process:

Either ID or Value, or both ID and Value can be passed to an API. But, when both the values are passed, ID based parameters takes precedence over value based parameters, that is, if both parameters are passed, the value based parameter is ignored and the ID based parameter is used.

When both the Value and ID of an attribute are passed to an API, a message is generated to inform the API caller that some input has been ignored. This is not an error message. The API continues with its regular processing.

Each value has to resolve into one ID. Failure to resolve a value into an ID results in an error and is associated with an error message. The API aborts processing and returns with an error return status.

4

Business Objects

Item Cost

Item Cost specifies cost of a given Item. To understand how it is constructed, let us take an example that Item - A is made up of Item - B and Item - C.

Now, cost of an Item - A = Material Cost of an Item - B + Material Cost of an Item - C + Cost of Manufacturing Item - A from Item - B and Item - C

To better differentiate these costs, business object "Item Cost" is formatted in the following way:

Cost of an Item - A

- This Level Cost Detail (Cost of Manufacturing Item - A)
- Machine Cost, Labor Cost, etc.
- Lower Level Cost Detail (Material Cost of an Item - B + Material Cost of an Item - C)
- Material Cost of components.

Business object Item Cost is made up of 3 entities - Cost Header, This Level Cost Details, and Lower Level Cost Details.

Cost Header entity specifies Calendar, Period, Warehouse, Cost Method, and Item for which cost is defined. This Level Cost Detail entity specifies Cost Component Class and Analysis Code and exact cost in figures. Similarly, Lower Level Cost Detail entity specifies Cost Component Class and Analysis Code and exact cost in figures.

Cost Header Entity

The Cost header entity specifies attributes which identify what this cost information is about at the Parent Level. It specifies the following attributes of business object - Item Cost:

- Calendar
- Period
- Cost Method
- Warehouse
- Item

Following is the definition of Header_Rec_Type:

Attribute	Required	Description
Calendar_code	Y	A valid Calendar Code must exist in cm_cldr_mst.
period_code	Y	A valid Period Code must exist in cm_cldr_dtl and must not be closed. For UPDATE operation, the period must be open. If the period is open, then allow all the operations. If the period is frozen, then only insert is allowed. No updates and deletes are allowed.
cost_mthd_code	Y	A valid Cost Method Code must exist in cm_mthd_mst.
whse_code	Y	A valid Warehouse Code must exist in ic_whse_mst.
item_id	N	Valid Item ID (Either item_id or item_no is required). Item ID/Item No must exist in ic_item_mst.
item_no	N	Valid Item No. (Either item_id or item_no is required). Item ID/Item No must exist in ic_item_mst.
user_name	Y	Valid Oracle Application's user name. User Name must exist in fnd_user.

This Level Cost Detail Entity

This Level Cost Detail entity specifies exact cost figure of a given cost component and analysis code. Along with that it also specifies other details which is specific to OPM. Attributes of This Level Cost Detail entity are:

- Cost Component Class
- Analysis Code
- Component Cost
- Burden Indicator
- Rollover Indicator
- Total Quantity
- Cost Calculation Origin
- Raw Material Calculation Type
- Delete Mark
- Descriptive Flexfield Segment1 to Segment30
- Attribute Category

Following is the definition of This_Level_Dtl_Tbl_Type:

Attribute	Required	Description
cmpntcost_id	N	In case of Insert, this is generated for each new record. Otherwise, the value supplied in this parameter is used to locate unique cost detail record.
cost_cmpntcls_id	N	Valid Cost Component Class ID (Either cost_ cmpntcls_id or cost_cmpntcls_code is required). Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst.
cost_cmpntcls_code	N	Valid Cost Component Class Code (Either cost_ cmpntcls_id or cost_cmpntcls_code is required). Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst.
cost_analysis_code	Y	Valid Cost Analysis Code. Cost Analysis Code must exist in cm_alys_mst.
cmpnt_cost	Y	Component Cost must be a valid number.

Attribute	Required	Description
burden_ind	Y	Burden Indicator can be either 0 or 1. If Burden Indicator is 1, then Cost Component Class must have usage indicator = 2.
total_qty	N	Total transaction quantity used in calculating actual cost
costcalc_orig	N	This field is only a placeholder and value supplied by API caller is not considered. For each new record created, this field is set to 3 to indicate API origin.
Rmcalc_type	N	Raw Material Cost Calculation Type (Applicable only under actual costs, otherwise it is =0 1=PMAC, 2=PWAC, 3=PPAC, 4=LSTT, 5=LSTI
Delete_mark	Y	Delete Mark must be either 0 or 1. Insert API ignores the value sent. New records are always created with Delete Mark set to 0. Record cannot be deleted using Update API (marked for purge), but can be undeleted.
attribute1-30	N	Descriptive Flexfield Segment. Attribute1 through Attribute30 and Attribute Category are optional and can have alphanumeric data.
attribute_category	N	Descriptive Flexfield Segment. Attribute1 through Attribute30 and Attribute Category are optional and can have alphanumeric data.

Lower Level Cost Detail Entity

Lower Level Cost Detail entity specifies exact cost figure of a given cost component and analysis code at lower level. Its attributes are:

- Cost Component Class
- Analysis Code
- Component Cost

Following is the definition of Lower_Level_Dtl_Tbl_Type:

Attribute	Required	Description
cmpntcost_id	N	In case of Insert, this is generated for each new record. Otherwise, value supplied in this parameter is used to locate unique cost detail record.
cost_cmpntcls_id	N	Valid Cost Component Class ID (Either cost_ cmpntcls_id or cost_cmpntcls_code is required
cost_cmpntcls_code	N	Valid Cost Component Class Code (Either cost_ cmpntcls_id or cost_cmpntcls_code is required
cost_analysis_code	Y	Valid Cost Analysis Code. Cost Analysis Code must exist in cm_alys_mst.
cmpnt_cost	Y	Component Cost must be a valid number.
delete_mark	Y	Standard 0=Active Record (Default), 1=Marked for (logical) Deletion.

Parameter - x_costcmpnt_ids (OUT)

This table type parameter has multiple records of newly generated surrogates for cost component records. API Caller receives this OUT parameter with cost component IDs for each cost record both at this level and the lower level. The following table explains how these IDs are returned:

Attribute	Description
cost_cmpntcls_id	Cost Component Class ID.
cost_analysis_code	Cost Analysis Code.
cost_level	Cost Level where the record is created. Valid values are 0=This Level, 1=Lower Level.
cmpntcost_id	Surrogate generated for combination of Cost Component Class ID+ Cost Analysis Code+Cost Level.

Structure and Logic

This section explains structure and logic of public APIs for Item cost.

Package Specification - GMFPCSTS.pls

This file holds definition of all the variables, procedures, and functions that are available to users.

```
TYPE header_rec_type IS RECORD
(
  calendar_code    cm_cmpt_dtl.calendar_code%TYPE :=
, period_code      cm_cmpt_dtl.period_code%TYPE   :=
, cost_mthd_code   cm_cmpt_dtl.cost_mthd_code%TYPE:=
, whse_code        cm_cmpt_dtl.whse_code%TYPE     :=
, item_id          NUMBER                         :=
, item_no          ic_item_mst.item_no%TYPE       :=
, user_name        fnd_user.user_name%TYPE        :=
);

TYPE this_level_dtl_rec_type IS RECORD
(
  cmpntcost_id       NUMBER                       :=
, cost_cmpntcls_id   NUMBER                       :=
, cost_cmpntcls_code cm_cmpt_mst.cost_cmpntcls_code%TYPE:=
, cost_analysis_code cm_cmpt_dtl.cost_analysis_code%TYPE:=
, cmpnt_cost         NUMBER                       :=
, burden_ind         NUMBER                       :=
, total_qty          NUMBER                         :=
, costcalc_orig      NUMBER                         :=
, rmcalc_type        NUMBER                         :=
, delete_mark        NUMBER                         :=
, attribute1         cm_cmpt_dtl.attribute1%TYPE    :=
 ...
, attribute30        cm_cmpt_dtl.attribute25%TYPE   :=
, attribute_category cm_cmpt_dtl.attribute_category%TYPE:=
 );

TYPE this_level_dtl_tbl_type IS TABLE OF this_level_dtl_rec_type
                        INDEX BY BINARY_INTEGER;

TYPE lower_level_dtl_rec_type IS RECORD
(
  cmpntcost_id      NUMBER                         :=
, cost_cmpntcls_id  NUMBER                         :=
, cost_cmpntcls_code cm_cmpt_mst.cost_cmpntcls_code%TYPE:=
, cost_analysis_code cm_cmpt_dtl.cost_analysis_code%TYPE:=
, cmpnt_cost        NUMBER                         :=
, delete_mark       NUMBER                         :=
 );

TYPE lower_level_dtl_tbl_type IS TABLE OF lower_level_dtl_rec_type
                        INDEX BY BINARY_INTEGER;

TYPE costcmpnt_ids_rec_type IS RECORD
(
  cost_cmpntcls_id  NUMBER                         :=
, cost_analysis_code cm_cmpt_dtl.cost_analysis_code%TYPE:=
, cost_level        NUMBER                         :=
```

```
, cmpntcost_id      NUMBER                          :=
);

TYPE costcmpnt_ids_tbl_type IS TABLE OF costcmpnt_ids_rec_type
                         INDEX BY BINARY_INTEGER;
PROCEDURE Create_Item_Cost
(         p_api_versionIN  NUMBER ,
p_init_msg_listIN  VARCHAR2 := FND_API.G_FALSE,
p_commit  IN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Header_Rec_Type,
p_this_level_dtl_tblIN  This_Level_Dtl_Tbl_Type,
p_lower_level_dtl_TblIN  Lower_Level_Dtl_Tbl_Type,

x_costcmpnt_idsOUT costcmpnt_ids_tbl_type

);

PROCEDURE Update_Item_Cost
(         p_api_versionIN  NUMBER ,
p_init_msg_listIN  VARCHAR2 := FND_API.G_FALSE,
p_commit  IN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Header_Rec_Type,
p_this_level_dtl_tblIN  This_Level_Dtl_Tbl_Type,
p_lower_level_dtl_TblIN  Lower_Level_Dtl_Tbl_Type

);

PROCEDURE Delete_Item_Cost
(         p_api_versionIN  NUMBER ,
p_init_msg_listIN  VARCHAR2 := FND_API.G_FALSE,
p_commit  IN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Header_Rec_Type,
p_this_level_dtl_tblIN  This_Level_Dtl_Tbl_Type,
p_lower_level_dtl_TblIN  Lower_Level_Dtl_Tbl_Type
```

```
);

PROCEDURE Get_Item_Cost
(         p_api_versionIN  NUMBER ,
p_init_msg_listIN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Header_Rec_Type,

x_this_level_dtl_tblOUT This_Level_Dtl_Tbl_Type,
x_lower_level_dtl_TblOUT Lower_Level_Dtl_Tbl_Type

);
```

Public Procedure Create_Item_Cost

Procedure Create_Item_Cost is used to insert component cost details of an Item. API caller needs to supply the data through the following three parameters :

p_header_rec Header level Information for Item Cost record

p_this_level_dtl_tbl This level cost details

p_lower_level_dtl_tbl Lower level cost details

> **Note:** For details on the above parameters refer to the *Structure and Logic* topic in this guide.

The Public API performs all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. After finishing validations on input data, the public API inserts component cost details by calling necessary routines. The API generates surrogate key - cmpntcost_id for each component cost detail begin inserted. Also, the API returns a parameter with cmpntcost_ids for each newly created component cost record.

Note: During the validation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

During the insert (after the validations to all detail records) if insert fails, then all the detail records for the item in process are not inserted. An error message displays with the details of the record for which insert failed.

Only one message is given for number of rows successfully inserted for an item.

Public Procedure Update_Item_Cost

Procedure Update_Item_Cost is used to update this level component cost of an item. To update component cost of an item, API caller needs to specify attributes which uniquely classify the individual record to update. These attributes are supplied through following two parameters along with the exact cost which needs to be updated:

p_header_rec Header level Information for Item Cost record

p_this_level_dtl_tbl This level cost details

p_lower_level_dtl_tbl Lower level cost details

Note: For details on the above parameters refer to the *Structure and Logic* topic in this guide.

The second and third parameter holds surrogate - cmpntcost_id for the Item Cost record. If a valid value is supplied for this attribute, then it is used to identify the record to be updated; otherwise, the system looks at calendar, period, cost method code, warehouse code, and item id in the p_header_rec parameter as well as cost component class and analysis code in the p_this_level_dtl_tbl parameter to identify which record needs to be updated. The same is true for p_lower_level_dtl_tbl records.

Note: During the validation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

If the record is not found using either cmpntcost_id or using unique key, then all the records for that item are not updated. An error message displays with the details of the record for which update failed.

Only one message is given for number of rows successfully updated for an item.

If any column must be updated to NULL, then you must pass in FND_API.G_MISS_CHAR, FND_API.G_MISS_NUM, or FND_API.G_MISS_DATE variables to the API column values to update the column to NULL in the database.

Public Procedure Delete_Item_Cost

Procedure Delete_Item_Cost is used to delete this level component cost of an item. To delete component cost of an item, API caller needs to specify attributes which uniquely classify the individual record to delete. These attributes are supplied through the following parameters along with the exact cost which needs to be deleted:

p_header_rec Header level Information for Item Cost record

p_this_level_dtl_tbl This level cost details

p_lower_level_dtl_tbl Lower level cost details

Note: For details on these parameters, refer to the *Structure and Logic* topic in this guide.

The second and third parameter holds surrogate - cmpntcost_id for the Item Cost record. If valid value is supplied for this attribute, then it is used to identify the record to be deleted; otherwise the system looks at calendar, period, cost method code, warehouse code, and item id in the p_header_rec parameter as well as cost component class and analysis code in the p_this_level_dtl_tbl parameter to identify

which record needs to be deleted. The same is true for p_lower_level_dtl_tbl records.

Note: During the validation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

If the record is not found using either cmpntcost_id or using unique key, then all the records for that item are not deleted. An error message is shown with the details of the record for which delete failed.

Only one message is given for number of rows successfully deleted for an item.

Public Procedure Get_Item_Cost

Procedure Get_Item_Cost is used to retrieve the cost of an item. To retrieve component costs of an item, API caller needs to specify attributes which uniquely classify which item costs needs to retrieve. These attributes are supplied through the following parameter:

p_header_rec Header level Information for Item Cost record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Item Costs are returned back to the caller using the following OUT parameters:

x_this_level_dtl_tbl This level cost details

x_lower_level_dtl_tbl Lower level cost details

Resource Cost

The Resource Cost API specifies cost of a given resource per unit of usage. This cost is added to a product whenever it uses this resource in its manufacturing. Resource cost is defined for an organization, for calendar - period and for a given cost method.

Resource Cost Entity

Resource Cost maps to a single entity called Resource Cost. Following are the attributes of this entity:

- Resource
- Organization
- Calendar
- Period
- Cost Method
- Usage Unit of Measure
- Nominal Cost
- Delete Mark

Following is the definition of Resource_Cost_Rec_Type:

Attribute	Required	Description
resources	Y	Valid Resource Code must exist in cr_rsrc_mst
orgn_code	Y	Valid Organization Code must exist in sy_orgn_mst
calendar_code	Y	Valid Calendar Code must exist in cm_cldr_mst.
period_code	Y	Valid Period code must exist in cm_cldr_dtl and must not be closed
cost_mthd_code	Y	Valid Cost Method Code must exist in cm_mthd_mst
usage_um	Y	Valid Unit of Measure must exist in sy_uoms_mst
nominal_cost	Y	Valid Number must be a number greater than 0

Attribute	Required	Description
delete_mark	N	Either 0 or 1. Default is 0
user_name	Y	Valid Oracle Application's user name. User name must exist in fnd_user

Structure and Logic

This section explains structure and logic of public APIs for Resource Cost.

Package Specification - GMFPRESS.pls

This file holds definition of all the variables, procedures, and functions that are available.

```
TYPE Resource_Cost_Rec_Type IS RECORD
(  resources cm_rsrc_dtl.resources%TYPE        :=  ,
   orgn_code cm_rsrc_dtl.orgn_code%TYPE        :=  ,
   calendar_code cm_rsrc_dtl.calendar_code%TYPE:=  ,
   period_code  cm_rsrc_dtl.period_code%TYPE   :=  ,
   cost_mthd_code cm_rsrc_dtl.cost_mthd_code%TYPE:=  ,
   usage_um   cm_rsrc_dtl.usage_um%TYPE        :=  ,
   nominal_cost  NUMBER                        :=   ,
   delete_mark  cm_rsrc_dtl.delete_mark%TYPE   := ,
   user_name   fnd_user.user_name%TYPE         :=
 );

      PROCEDURE Create_Resource_Cost
      (p_api_versionIN  NUMBER ,
p_init_msg_list              IN  VARCHAR2 := FND_API.G_FALSE,
p_commitIN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN Resource_Cost_Rec_Type
      );

      PROCEDURE Update_Resource_Cost
      (p_api_versionIN  NUMBER ,
p_init_msg_list              IN  VARCHAR2 := FND_API.G_FALSE,
p_commitIN  VARCHAR2 := FND_API.G_FALSE,
```

```
x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN Resource_Cost_Rec_Type
      );

      PROCEDURE Delete_Resource_Cost
      (p_api_versionIN  NUMBER ,
p_init_msg_list               IN  VARCHAR2 := FND_API.G_FALSE,
p_commitIN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN Resource_Cost_Rec_Type
      );

      PROCEDURE Get_Resource_Cost
      (p_api_versionIN  NUMBER ,
p_init_msg_list               IN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN  Resource_Cost_Rec_Type,
x_resource_cost_recOUT Resource_Cost_Rec_Type
      );
```

Public Procedure Create_Resource_Cost

Procedure Create_Resource_Cost is used to insert resource cost details of an item. API caller needs to supply the data through the following parameter :

p_resource_cost_rec Information for Resource Cost record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

The Public API performs all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. After finishing validations on input data, the public API inserts resource cost details by calling necessary routines.

Public Procedure Update_Resource_Cost

Procedure Update_Resource_Cost is used to update nominal cost of a resource. To update nominal cost of a resource, API caller needs to specify attributes which uniquely classify the individual record to update. These attributes are supplied through the following parameter:

p_resource_cost_rec Information for Resource Cost record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

If any column must be updated to NULL, then you must pass in FND_API.G_MISS_CHAR, FND_API.G_MISS_NUM or FND_API.G_MISS_DATE variables to the API column values to update the column to NULL in the database.

Public Procedure Delete_Resource_Cost

Procedure Delete_Resource_Cost is used to delete a particular Resource Cost record from OPM. To delete a Resource Cost record, API caller needs to specify attributes which uniquely classify the individual record to delete. These attributes are supplied through the following parameters:

p_resource_cost_rec Information for Resource Cost record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Public Procedure Get_Resource_Cost

Procedure Get_Resource_Cost is used to retrieve the cost of a Resource. To retrieve Resource Cost information, the API caller needs to specify attributes which

uniquely classify which Resource costs needs to retrieve. These attributes are supplied through the following parameters:

p_resource_cost_rec Information for Resource Cost record

> **Note:** For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Resource Costs are returned back to caller using the following OUT parameters:

x_resource_cost_rec Information for Resource Cost record

Allocation Definition

Allocation Definition specifies information for an allocation code about the items to which expenses are allocated, the allocation criteria based upon the basis account or fixed percent, and the cost component class bucket to which the allocated cost goes.

Allocation Definition Entity

Allocation Definition maps to single entity called Allocation Definition. Following are the attributes of this entity:

- Allocation Code
- Allocation Method
- Item No
- Basis Account Key
- Balance Type
- Basis YTD PTD
- Fixed Percent
- Cost Component Class
- Analysis Code
- Warehouse Code
- Delete Mark

Following is the definition of Allocation_Def_Rec_Type:

Attribute	Required	Description
alloc_id	Y	Surrogate for Allocation Code (Either alloc_id OR co_code + alloc_code is required.)
co_code	Y	Valid Company Code (Either alloc_id OR co_ code + alloc_code is required.)
alloc_code	Y	Valid Allocation Code (Either alloc_id OR co_ code + alloc_code is required.)
alloc_method	Y	Valid Allocation Method (Either 0 or 1)
line_no	N	Surrogate for detail record. Only used in case of Update operation.

Attribute	Required	Description
item_id	Y	Surrogate for Item No. (At least one of these two parameters is required.)
item_no	Y	Valid Item No. (At least one of these two parameters is required.)
Basis_account_key	Y	A valid Basis Account Key (Used only when Allocation Method = 0).
balance_type	Y	Balance Type must be 0 = Statistical; 1 = Budget; or 2 = Actual. (Used only when Allocation Method = 0).
bas_ytd_ptd	Y	Basis YTP/PTD must be either 0 = Period To Date Basis Amount or 1 = Year To Date Basis Amount. (Used only when Allocation Method = 0).
fixed_percent	Y	Fixed percentage must be a valid number between 1 and 100 (Used only when Allocation Method = 1).
cmpntcls_id	Y	Surrogate for Cost Component Class (At least one of these two parameters is required.)
cost_cmpntcls_code	Y	Valid Cost Component Class (At least one of these two parameters is required.)
analysis_code	Y	Valid Analysis Code must exist in cm_alys_mst
whse_code	Y	Valid Warehouse Code must exits in ic_whse_ mst.
delete_mark	N	Either 0 or 1. Default is 0
user_name	Y	Valid Oracle Application's user name. User name must exists in fnd_user.

Structure and Logic

This section explains structure and logic of public APIs for Allocation Definition.

Package Specification - GMFPALCS.pls

This file holds definition of all the variables, procedures, and functions that are available.

```
TYPE Allocation_Definition_Rec_Type IS RECORD
(
alloc_id          NUMBER                          :=   ,
```

```
alloc_code        gl_aloc_mst.alloc_code%TYPE      :=  ,
co_code       sy_orgn_mst.co_code%TYPE             :=  ,
alloc_method      NUMBER                           :=   ,
line_no           NUMBER                           :=   ,
item_id           NUMBER                           :=   ,
item_no           ic_item_mst.item_no%TYPE         :=  ,
basis_account_key gl_aloc_bas.basis_account_key%TYPE:=  ,
balance_type     NUMBER                            :=   ,
bas_ytd_ptd      NUMBER                            :=   ,
fixed_percent    NUMBER                            :=   ,
cmpntcls_id      NUMBER                            :=   ,
cost_cmpntcls_code cm_cmpt_mst.cost_cmpntcls_code%TYPE :=  ,
analysis_code   cm_alys_mst.cost_analysis_code%TYPE :=  ,
whse_code       gl_aloc_bas.whse_code%TYPE         :=  ,
delete_mark      gl_aloc_bas.delete_mark%TYPE      :=              ,
user_name        fnd_user.user_name%TYPE           :=
 );

PROCEDURE Create_Allocation_Definition
(        p_api_versionIN  NUMBER ,
p_init_msg_list                  IN  VARCHAR2 := FND_API.G_FALSE,
p_commit IN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_allocation_definition_rec      IN  Allocation_Definition_Rec_Type
);

PROCEDURE Update_Allocation_Definition
(        p_api_versionIN  NUMBER ,
p_init_msg_list                  IN  VARCHAR2 := FND_API.G_FALSE,
p_commit IN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_allocation_definition_rec      IN Allocation_Definition_Rec_Type
);

PROCEDURE Delete_Allocation_Definition
(        p_api_versionIN  NUMBER ,
p_init_msg_list                  IN  VARCHAR2 := FND_API.G_FALSE,
```

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

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_allocation_definition_rec      IN Allocation_Definition_Rec_Type
);
```

Public Procedure Create_Allocation_Definition

Procedure Create_Allocation_Definition is used to insert allocation definition details. API caller needs to supply the data through the following parameter :

p_allocation_definition_rec Information for Allocation Definition record

> **Note:** For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

The Public API performs all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. After finishing validations on input data, the public API inserts allocation definition details by calling necessary routines.

Public Procedure Update_Allocation_Definition

Procedure Update_Allocation_Definition is used to update existing allocation definition record in OPM. To update allocation definition, API caller needs to specify attributes which uniquely classify the individual record to update. These attributes are supplied through the following parameter:

p_allocation_definition_rec Information for Allocation Definition record

> **Note:** For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Following are the attributes used to uniquely identify which record to update:

- ALLOC_ID or ALLOC_CODE + CO_CODE
- LINE_NO

If any column must be updated to NULL, then you must pass in FND_API.G_MISS_CHAR, FND_API.G_MISS_NUM or FND_API.G_MISS_DATE variables to the API column values to update the column to NULL in the database.

Public Procedure Delete_Allocation_Definition

Procedure Delete_Allocation_Definition is used to delete a particular Allocation Definition record from OPM. To delete an Allocation Definition record, the API caller needs to specify attributes which uniquely classify the individual record to delete. These attributes are supplied through the following parameters:

p_allocation_definition_rec Information for Allocation Definition record

> **Note:** For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Following are the attributes used to uniquely identify which record to delete:

- ALLOC_ID or ALLOC_CODE + CO_CODE
- LINE_NO

Burden Details

The Burden Details API is used to setup and maintain standard resource burdens. A burden is a cost associated with a resource other than the resource usage assigned in the routing. Burden Cost is added to the product cost whenever it uses the resource with which burden is associated. Burden is defined for an item, organization, warehouse, for calendar - period and for a given cost method.

Header Entity

Burden Details Header entity specifies attributes which identify what this burden details information is about at parent level. It specifies the following attributes of business object - Burden Details:

- Organization
- Item
- Warehouse
- Calendar
- Period
- Cost Method

Following is the definition of Header_Rec_Type:

Attribute	Required	Description
orgn_code	Y	Valid Organization Code must exist in sy_orgn_mst
item_id	N	Valid Item ID must exist in ic_item_mst (Either item_id or item_no is required)
item_no	N	Valid Item No. must exist in ic_item_mst (Either item_id or item_no is required)
whse_code	Y	Valid Warehouse Code must exist in ic_whse_mst
Calendar_code	Y	Valid Calendar Code must exist in cm_cldr_mst
period_code	Y	Valid Period Code must exist in cm_cldr_dtl
cost_mthd_code	Y	Valid Cost Method Code must exist in cm_mthd_mst
user_name	Y	Valid Oracle Application's user name. User Name must exists in fnd_user.

Detail Entity

The Detail Entity API specifies exact resource with which burden is associated along with component, analysis code, burden usage, item quantity and item UOM. It also specifies lot other details which is very specific to OPM. Attributes of Burden Details - Detail Entity entity are:

- Resource
- Cost Component Class
- Analysis Code
- Burden Usage
- Item Quantity
- Item UOM
- Burden Quantity
- Burden UOM
- Base Currency Code
- Delete Mark

Following is the definition of This_Level_Dtl_Tbl_Type:

Attribute	Required	Description
burdenline_id	N	For Insert, this is generated for each new record. Otherwise, the value supplied in this parameter is used to locate unique burden detail record.
resources	Y	Valid Resource Code must exist in cr_rsrc_mst.
cost_cmpntcls_id	N	Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst and must have usage indicator = 2 (Burden/Overhead).
cost_cmpntcls_ code	N	Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst and must have usage indicator = 2 (Burden/Overhead).
cost_analysis_code	Y	Valid Cost Analysis Code must exist in cm_alys_ mst.
burden_usage	Y	Valid Number between 0.000000001 and 999999999.999999999.

Attribute	Required	Description
item_qty	Y	Valid Number between 0.000000001 and 999999999.999999999.
item_um	Y	Item UOM must be a valid Unit of Measure. Since this Item UOM can be any UOM, a conversion factor exists for this UOM and the items actual UOM.
burden_qty	Y	Valid Number between 0.000000001 and 999999999.999999999.
burden_um	Y	Burden UOM must be a valid Unit of Measure and Burden UOM type must be the same as Resource UOM Type.
Delete_mark	Y	Delete Mark must be either 0 or 1. Insert API ignores the value sent. New records are created with Delete Mark set to 0. Record cannot be deleted using Update API (marked for purge), but can be undeleted.

Parameter - x_burdenline_ids (OUT)

This is a table type parameter having multiple records of newly generated surrogates for burden detail records. The API Caller receives this OUT parameter with Burden Line IDs for each Burden Detail record. The following table explains how these IDs are returned:

Attribute	Description
Resources	Resource used for the burden cost.
cost_cmpntcls_id	Cost Component Class ID.
cost_analysis_code	Cost Analysis Code.
burdenline_id	Surrogate generated for combination of Resource+Cost Component Class ID+ Cost Analysis Code.

Structure and Logic

This section explains structure and logic of public APIs for Burden Details.

Package Specification - GMFPBRDS.pls

This file holds definition of all the variables, procedures, and functions that are available.

```
TYPE Burden_Header_Rec_Type IS RECORD
(
orgn_code     sy_orgn_mst.orgn_code%TYPE        :=  ,
item_id       NUMBER                            :=   ,
item_no       ic_item_mst.item_no%TYPE          :=  ,
whse_code     cm_brdn_dtl.whse_code%TYPE        :=  ,
calendar_code cm_brdn_dtl.calendar_code%TYPE    :=  ,
period_code   cm_brdn_dtl.period_code%TYPE      :=  ,
cost_mthd_code cm_brdn_dtl.cost_mthd_code%TYPE  :=  ,
user_name     fnd_user.user_name%TYPE           :=
);

 TYPE Burden_Dtl_Rec_Type IS RECORD
 (
burdenline_id   NUMBER                          :=   ,
resources      cr_rsrc_mst.resources%TYPE       :=  ,
cost_cmpntcls_id NUMBER                         :=   ,
cost_cmpntcls_codecm_cmpt_mst.cost_cmpntcls_code%TYPE:=,
cost_analysis_code cm_alys_mst.cost_analysis_code%TYPE:=,
burden_usage   NUMBER                           :=   ,
item_qty       NUMBER                           :=   ,
item_um        cm_brdn_dtl.item_um%TYPE         :=  ,
burden_qty     NUMBER                           :=   ,
burden_um      cm_brdn_dtl.item_um%TYPE         :=  ,
burden_factor  NUMBER                           :=   ,
delete_mark    cm_brdn_dtl.delete_mark%TYPE     :=
 );

TYPE Burden_Dtl_Tbl_Type IS TABLE OF Burden_Dtl_Rec_Type
                        INDEX BY BINARY_INTEGER;

TYPE Burdenline_Ids_Rec_Type IS RECORD
(
resources     cm_rsrc_dtl.resources%TYPE        :=  ,
cost_cmpntcls_id NUMBER                           :=   ,
cost_analysis_code cm_brdn_dtl.cost_analysis_code%TYPE:=,
burdenline_id NUMBER                            :=
);

TYPE Burdenline_Ids_Tbl_Type IS TABLE OF Burdenline_Ids_Rec_Type
```

```
                        INDEX BY BINARY_INTEGER;
PROCEDURE Create_Burden_Details
(       p_api_versionIN  NUMBER ,
p_init_msg_list                 IN  VARCHAR2 := FND_API.G_FALSE,
p_commitIN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Burden_Header_Rec_Type,
p_dtl_tblIN  Burden_Dtl_Tbl_Type,

x_burdenline_idsIN  Burdenline_Ids_Tbl_Type
);

PROCEDURE Update_Burden_Details
(       p_api_versionIN  NUMBER ,
p_init_msg_list                 IN  VARCHAR2 := FND_API.G_FALSE,
p_commitIN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Burden_Header_Rec_Type,
p_dtl_tblIN  Burden_Dtl_Tbl_Type,
);

PROCEDURE Delete_Burden_Details
(       p_api_versionIN  NUMBER ,
p_init_msg_list                 IN  VARCHAR2 := FND_API.G_FALSE,
p_commitIN  VARCHAR2 := FND_API.G_FALSE,

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Burden_Header_Rec_Type,
p_dtl_tblIN  Burden_Dtl_Tbl_Type,
);

PROCEDURE Get_Item_Cost
(       p_api_versionIN  NUMBER ,
p_init_msg_list                 IN  VARCHAR2 := FND_API.G_FALSE,
```

```
x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_header_recIN  Burden_Header_Rec_Type,

x_dtl_tblOUT Burden_Dtl_Tbl_Type
);
```

Public Procedure Create_Burden_Details

Procedure Create_Burden_Details is used to insert standard resource burdens. API caller supplies the data through the following parameter:

p_header_rec Header level Information for Burden Details record

p_dtl_tbl Detail level Information for Burden Details record

> **Note:** For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

The Public API performs all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. After finishing validations on input data, the public API inserts burden details by calling necessary routines. The API generates surrogate key - burdenline_id for each burden details row begin inserted. Also, the API returns a parameter with burdenline_ids for each newly created burden detail record.

Note: During the validation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

During the insert (after the validations to all detail records) if insert fails for any reason, then all the detail records for the item in process are not inserted. An error message displays with the details of the record for which insert failed.

Only one message is given for number of rows successfully inserted for an item.

Public Procedure Update_Burden_Details

Procedure Update_Burden_Details is used to update this level component cost of an item. To update component cost of an item, API caller needs to specify attributes which uniquely classify the individual record to update. These attributes are supplied through the following parameters along with the other details to update:

p_header_rec Header level Information for Burden Details record

p_dtl_tbl Detail level Information for Burden Details record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

The second parameter holds surrogate - burdenline_ids for the Burden Cost record. If a valid value is supplied for this attribute, then it is used to identify the record to be updated; otherwise, the system looks at organization, item, warehouse, calendar, period, and cost method code in the p_header_rec parameter as well as resources, cost component class and analysis code in the p_dtl_tbl parameter to identify which record to update.

Note: During the validation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

If the record is not found using either cmpntcost_id or using unique key, then all the records for that item are not updated. An error message displays with the details of the record for which update failed.

Only one message is given for number of rows successfully updated.

If any column must be updated to NULL, then you must pass in FND_API.G_MISS_CHAR, FND_API.G_MISS_NUM or FND_API.G_MISS_DATE variables to the API column values to update the column to NULL in the database.

Public Procedure Delete_Burden_Details

Procedure Delete_Burden_Details is used to delete burden details. To delete burden details, API caller needs to specify attributes which uniquely classify the individual record to be deleted. These attributes are supplied through the following parameters along with the exact cost to delete:

p_header_rec Header level Information for Burden Details record

p_dtl_tbl Detail level Information for Burden Details record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

The second parameter holds surrogate - burdenline_id for the burden details record. If a valid value is supplied for this attribute, then it is used to identify the record to delete; otherwise, the system looks at organization, item, warehouse, calendar, period, and cost method code in the p_header_rec parameter as well as resources, cost component class and analysis code in the p_dtl_tbl parameter to identify which record to delete.

Note: During the validation process, if a detail record fails to go through all the validations, then only that particular detail record is skipped or eliminated and the process continues with the next detail record.

If the record is not found using either cmpntcost_id or using unique key, then all the records for that item are not deleted. An error message displays with the details of the record for which delete failed.

Only one message is given for number of rows successfully deleted.

Public Procedure Get_Burden_Details

Procedure Get_Burden_Details is used to retrieve burden details. To retrieve burden details, API caller needs to specify attributes which uniquely classify which burden details needs to retrieve. These attributes are supplied through the following parameter:

p_header_rec Header level Information for Burden Details record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Burden Details are returned back to caller using the following OUT parameters:

x_dtl_tbl Burden details

Lot Cost Adjustments

Lot Cost Adjustments adjust the final calculated lot cost of a raw material or product based on the unit cost. Lot costs are recalculated based on the adjustments entered for the specified company, item, warehouse, lot, sublot, and adjustment date.

Header Entity

The Lot Cost Adjustment header entity specifies attributes that identify cost information at the parent level. It specifies the following attributes for the business object Lot Cost Adjustment:

- Company
- Lot Cost Method
- Item
- Warehouse
- Lot
- Sublot
- Adjustment Date
- Reason Code
- Attributes 1-30
- Attribute Category
- Delete Mark

Parameter p_header_rec

This is a record type parameter having header level attributes for Lot Cost Adjustment. The API caller passes appropriate values for each of its attributes:

Attribute	Required	Description
Adjustment_id	N	Insert API sets this value using sequences. For Update and Delete API, if passed, then attributes are not required. Otherwise, other attributes are mandatory.
Co_code	Y	Valid Company Code.
cost_mthd_code	Y	Valid Cost Method Code.

Attribute	Required	Description
item_id	N1	Valid Item ID (Either item_id or item_no is required).
item_no	N1	Valid Item Number (Either item_id or item_no is required).
whse_code	Y	Valid Warehouse Code.
Lot_id	N2	Valid Lot ID (Either lot_id or lot_no is required).
Lot_no	N2	Valid Lot Number (Either lot_id or lot_no is required).
Sublot_no	N	Valid Sublot Number.
Adjustment_date	Y	Valid date and time.
user_name	Y	Valid Oracle Applications user name.
Delete_mark	Y	Delete Mark must be either 0 or 1. Insert API ignores the value sent. New records are always created with Delete Mark set to 0. Record cannot be deleted using Update API (marked for purge), but can be undeleted.
reason_code	Y	Valid Reason Code.
attribute1-30	N	Descriptive Flexfield Segment. Attribute1 through Attribute30 and Attribute Category are optional and can have alphanumeric data.
attribute_ category	N	Descriptive Flexfield Segment. Attribute1 through Attribute30 and Attribute Category are optional and can have alphanumeric data.

Detail Entity

The Lot Cost Adjustment Detail entity specifies the exact adjustment cost for a given cost component and analysis code. Attributes of Lot Cost Adjustment Detail entity are:

- Cost Component Id and Class
- Analysis Code
- Adjustment Cost
- Text Code
- Delete Mark

Parameter p_dtl_tbl

This is a table type parameter with multiple records of Lot Cost Adjustment details. The API caller passes appropriate values for each of its attributes:

Attribute	Required	Description
adjustment_ dtl_id	N1	Insert API sets this value using sequences. For Update and Delete API, if passed, then attributes are not required. Otherwise, other attributes are mandatory.
adjustment_id	N2	Valid Lot Cost Adjustment Header ID.
cost_cmpntcls_ id	N2	Valid Cost Component Class ID.
cost_cmpntcls_ code	N2	Valid Cost Component Class Code.
cost_analysis_ code	Y	Valid Cost Analysis Code.
adjustment_ cost	Y	Adjustment Cost.
text_code	Y	An ID that joins any rows of text in this table to the text table for this application.

Structure and Logic

This section explains the structure and logic of public APIs for Lot Cost Adjustments.

Package Specification - GMFPLCAS.pls

This file holds definitions of all the variables, procedures, and functions that are available.

```
CREATE OR REPLACE PACKAGE GMF_LotCostAdjustment_PUB AS
/* $Header: GMFPLCAS.pls 115.1 2004/04/16 21:05:21 anthiyag noship $ */

TYPE Lc_Adjustment_Header_Rec_Type
IS
RECORD
(
adjustment_idgmf_lot_cost_adjustments.adjustment_id%TYPE
, co_code              sy_orgn_mst.co_code%TYPE
, cost_mthd_code        cm_mthd_mst.cost_mthd_code%TYPE
, item_id              ic_item_mst.item_no%TYPE
```

```
, item_no              ic_item_mst.item_no%TYPE
, whse_code             ic_whse_mst.whse_code%TYPE
, lot_id               ic_lots_mst.lot_id%TYPE
, lot_no               ic_lots_mst.lot_no%TYPE
, sublot_no              ic_lots_mst.sublot_no%TYPE
, adjustment_date        DATE
, reason_code        cm_reas_cds.reason_code%TYPE
, delete_mark        gmf_lot_cost_adjustments.delete_mark%TYPE
, ATTRIBUTE1                    VARCHAR2(240)
, ATTRIBUTE2                    VARCHAR2(240)
, ATTRIBUTE3                    VARCHAR2(240)
, ATTRIBUTE4                    VARCHAR2(240)
, ATTRIBUTE5                    VARCHAR2(240)
, ATTRIBUTE6                    VARCHAR2(240)
, ATTRIBUTE7                    VARCHAR2(240)
, ATTRIBUTE8                    VARCHAR2(240)
, ATTRIBUTE9                    VARCHAR2(240)
, ATTRIBUTE10                   VARCHAR2(240)
, ATTRIBUTE11                   VARCHAR2(240)
, ATTRIBUTE12                   VARCHAR2(240)
, ATTRIBUTE13                   VARCHAR2(240)
, ATTRIBUTE14                   VARCHAR2(240)
, ATTRIBUTE15                   VARCHAR2(240)
, ATTRIBUTE16                   VARCHAR2(240)
, ATTRIBUTE17                   VARCHAR2(240)
, ATTRIBUTE18                   VARCHAR2(240)
, ATTRIBUTE19                   VARCHAR2(240)
, ATTRIBUTE20                   VARCHAR2(240)
, ATTRIBUTE21                   VARCHAR2(240)
, ATTRIBUTE22                   VARCHAR2(240)
, ATTRIBUTE23                   VARCHAR2(240)
, ATTRIBUTE24                   VARCHAR2(240)
, ATTRIBUTE25                   VARCHAR2(240)
, ATTRIBUTE26                   VARCHAR2(240)
, ATTRIBUTE27                   VARCHAR2(240)
, ATTRIBUTE28                   VARCHAR2(240)
, ATTRIBUTE29                   VARCHAR2(240)
, ATTRIBUTE30                   VARCHAR2(240)
, ATTRIBUTE_CATEGORY            VARCHAR2(30)
, user_name              fnd_user.user_name%TYPE
);

TYPE lc_adjustment_dtls_Rec_Type
IS
RECORD
```

```
(
adjustment_dtl_id gmf_lot_cost_adjustment_dtls.adjustment_dtl_id%TYPE
, adjustment_id         gmf_lot_cost_adjustment_dtls.adjustment_id%TYPE
, cost_cmpntcls_id     cm_cmpt_mst.cost_cmpntcls_id%TYPE
, cost_cmpntcls_code cm_cmpt_mst.cost_cmpntcls_code%TYPE
, cost_analysis_code cm_alys_mst.cost_analysis_code%TYPE
, adjustment_cost            gmf_lot_cost_adjustment_dtls.adjustment_cost%TYPE
, DELETE_MARK                   NUMBER(22)
, TEXT_CODE                     NUMBER(22)
);

TYPE lc_adjustment_dtls_Tbl_Type
IS
TABLE OF lc_adjustment_dtls_Rec_Type
INDEX BY BINARY_INTEGER;

PROCEDURE Create_LotCost_Adjustment
(
p_api_version IN  NUMBER
, p_init_msg_list IN  VARCHAR2 := FND_API.G_FALSE
, p_commit IN  VARCHAR2 := FND_API.G_FALSE
, x_return_status OUT NOCOPY VARCHAR2
, x_msg_count OUT NOCOPY NUMBER
, x_msg_data OUT NOCOPY VARCHAR2
, p_header_rec IN OUT NOCOPY Lc_Adjustment_Header_Rec_Type
, p_dtl_Tbl IN OUT NOCOPY Lc_adjustment_dtls_Tbl_Type
);

PROCEDURE Update_LotCost_Adjustment
(
p_api_version IN  NUMBER
, p_init_msg_list IN  VARCHAR2 := FND_API.G_FALSE
, p_commit IN  VARCHAR2 := FND_API.G_FALSE
, x_return_status OUT NOCOPY VARCHAR2
, x_msg_count OUT NOCOPY NUMBER
, x_msg_data OUT NOCOPY VARCHAR2
, p_header_rec IN OUT NOCOPY Lc_Adjustment_Header_Rec_Type
, p_dtl_Tbl IN OUT NOCOPY lc_adjustment_dtls_Tbl_Type
);

PROCEDURE Delete_LotCost_Adjustment
(
p_api_version IN  NUMBER
, p_init_msg_list IN  VARCHAR2 := FND_API.G_FALSE
, p_commit IN  VARCHAR2 := FND_API.G_FALSE
```

```
, x_return_statusOUT NOCOPY VARCHAR2
, x_msg_countOUT NOCOPY NUMBER
, x_msg_dataOUT NOCOPY VARCHAR2
, p_header_recIN OUT NOCOPY Lc_Adjustment_Header_Rec_Type
, p_dtl_TblIN OUT NOCOPY lc_adjustment_dtls_Tbl_Type
);

PROCEDURE Get_LotCost_Adjustment
(
p_api_versionIN  NUMBER
, p_init_msg_listIN  VARCHAR2 := FND_API.G_FALSE
, x_return_statusOUT NOCOPY VARCHAR2
, x_msg_countOUT NOCOPY NUMBER
, x_msg_dataOUT NOCOPY VARCHAR2
, p_header_recIN OUT NOCOPY Lc_Adjustment_Header_Rec_Type
, p_dtl_TblOUT NOCOPY lc_adjustment_dtls_Tbl_Type
);

END GMF_LotCostAdjustment_PUB ;
/
COMMIT;
EXIT;
```

Public Procedure Create_LotCost_Adjustment

Procedure Create_LotCost_Adjustment is used to insert Lot Cost Adjustments. API caller must supply the data through the following two parameters:

p_header_rec Header level Information for Lot Cost Adjustment record

p_dtl_tbl Lot Cost Adjustment details

> **Note:** For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

The Public API performs all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. After finishing validations on input data, public API inserts Lot Cost Adjustments by calling necessary routines.

Note: During the validation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

During the insert (after the validations to all detail records) if insert fails for any reason, then all the detail records for the item in process are not inserted. An error message displays with the details of the record for which insert failed.

Only one message is given for number of rows successfully inserted for an item.

Public Procedure Update_LotCost_Adjustment

Procedure Update_LotCost_Adjustment is used to update Adjustment cost and Text Code for an Company/Lot Cost Method/Item/lot/Sublot/Whse/Reason Code/Adjustment Date. If an adjustment exists and is already applied, then updates are not allowed. These attributes are supplied through the following parameters along with the exact cost to update:

p_header_rec Header level Information for Lot Cost Adjustment record

p_dtls_tbl Adjustment details

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

These parameters have primary key columns of header and detail tables. If a valid Adjustment ID is supplied, then it is used to identify the record to be updated. Otherwise, the system looks at company, cost method code, item, warehouse code, and lot in the p_header_rec parameter as well as header id, cost component class, and analysis code in the p_dtls_tbl parameter to identify which record must be updated.

If any column must be updated to NULL, then you must pass in FND_API.G_MISS_CHAR, FND_API.G_MISS_NUM or FND_API.G_MISS_DATE variables to the API column values to update the column to NULL in the database.

Note: During thevalidation process, if a detail record fails to go through all the validations, then the API stops the processing and returns an error status without finishing the remaining detail records.

If the record is not found, then all the records for that item are not updated. An error message displays with the details of the record for which update failed.

Only one message is given for number of rows successfully updated.

Public Procedure Delete_LotCost_Adjustment

Procedure Delete_LotCost_Adjustment is used to delete Lot Cost Adjustment details. If an adjustment exists and is already applied, deletes are not allowed. To delete the adjustment cost of an item, the API caller needs to specify attributes which uniquely classify the individual record being deleted. These attributes are supplied through the following two parameters along with the exact cost which needs to be deleted:

p_header_rec Header level Information for the Lot Cost Adjustment record

p_dtls_tbl Adjustment details

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

These parameters have primary key columns of header and detail tables. If a valid Adjustment ID is supplied, then it is used to identify the record to be updated. Otherwise, the system looks at company, lot cost method code, item, lot, sublot, warehouse, reason code and adjustment date and in the p_header_rec parameter as well as header id, cost component class, and analysis code in the p_dtls_tbl parameter to identify which record needs to be updated.

Note: During the validation process, if a detail record fails to go through all the validations, then only that particular detail record is skipped or eliminated and the process continues with the next detail record.

If the record is not found using either cmpntcost_id or using unique key, then all the records for that item are not deleted. An error message displays with the details of the record for which delete failed.

Only one message is given for number of rows successfully deleted.

Public Procedure Get_Lot_Cost_Adjustment

Procedure Get_Lot_Cost_Adjustment retrieves lot cost adjustments. To retrieve lot cost adjustments, API caller needs to specify attributes which uniquely classify which lot cost adjustments to retrieve. These attributes are supplied through the following parameter:

p_header_rec Header level Information for Lot Cost Adjustments record

Note: For details on the above parameters, refer to the *Structure and Logic* topic in this guide.

Lot Cost Adjustments are returned back to the caller using the following OUT parameters:

x_dtl_tbl Lot Cost Adjustments

A

Messages and Errors

This appendix covers:

- Handling Messages
- Interpreting Error Conditions
- Understanding Error Messages

Handling Messages

APIs put result messages into a message list. Programs calling APIs can then get the messages from the list and process them by either issuing them, loading them in a database table, or writing them to a log file.

Messages are stored in an encoded format to enable API callers to find out message names by using the standard functions provided by the message dictionary. It also allows storing these messages in database tables and reporting off these tables in different languages.

The structure of the message list is not public. Neither API developers nor API callers can access this list except through calling the API message utility routines mentioned below.

The following utility functions are defined in the FND_MSG_PUB package, in the file AFASMSGS.pls:

Initialize Initializes the API message list.

Add Adds a message to the API message list.

Get Gets a message from the API message list.

Count_Msg Returns the number of messages in the API message list.

Delete Deletes one or more messages from the API message list.

Reset Resets the index used in getting messages.

Count_And_Get Returns the number of messages in the API message list. If this number is one, it also returns the message data.

Refer to the documentation of these functions and procedures for usage information.

To add a message to the API message list, developers can use the regular message dictionary procedures FND_MESSAGE.SET_NAME and FND_MESSAGE.SET_TOKEN to set the message name and tokens on the message dictionary stack. Then call FND_MSG_PUB.Add to fetch the messages off the message dictionary stack and add it to the API message list.

To get a message from the API message list, API callers can use the procedure FND_MSG_PUB.Get. This procedure operates in the following modes:

First Gets the first message in the API message list

Next Gets the next message in the API message list

Last Gets the last message in the API message list

Previous Gets the previous message in the API message list

Specific Gets a specific message from the API message list

It is recommended that APIs provide their callers with the following information:

- message count
- message data

The message count holds the number of messages in the API message list. If this number is one, then message data holds the message in an encoded format.

```
x_msg_count                    OUT       NUMBER
x_msg_dataOUTVARCHAR2
```

Example:

```
PROCEDURE Create_OrderLine
```

```
( p_api_version           INNUMBER,
  p_init_msg_listINVARCHAR2 := FND_API.G_FALSE,
p_commit    IN  VARCHAR2 := FND_API.G_FALSE,
p_validation_levelIN  NUMBER:=
FND_API.G_VALID_LEVEL_FULL,

x_return_statusOUTVARCHAR2  ,
x_msg_countOUTNUMBER,
x_msg_dataOUTVARCHAR2 ,

p_line_recINLine_Rec_Type
)
IS
l_api_version           CONSTANT    NUMBER  :=  1.0;
l_api_name      CONSTANT    VARCHAR2(30):=  'Create_OrderLine';
BEGIN
--  Standard begin of API savepoint
    SAVEPOINTCreate_Line_PUB;
    -- Standard call to check for call compatibility.
    IF NOT FND_API.Compatible_API_Call ( l_api_version        ,
                    p_api_version        ,
                     l_api_name     ,
          G_PKG_NAME     )
THEN
RAISE FND_API.G_EXC_UNEXPECTED_ERROR;
END IF;
-- Check p_init_msg_list
IF FND_API.to_Boolean( p_init_msg_list ) THEN
FND_MSG_PUB.initialize;
END IF;
--  Initialize API return status to success
    x_return_status := FND_API.G_RET_STS_SUCCESS;

Validate_Line
(l_return_status,
l_line_rec
);

Price_Line
(l_return_status,
l_line_rec
);

Insert_Line
(l_return_status,
l_line_rec
```

```
);
IF FND_API.To_Boolean( p_commit ) THEN
COMMIT WORK;
END IF;
-- Get message count and if 1, return message data.
FND_MSG_PUB.Count_And_Get
    (  p_count         =>      x_msg_count     ,
        p_data          =>      x_msg_data
);
END Create_Line ;
```

Interpreting Error Conditions

The parameter x_return_status indicates whether the API was successful or failed. The values are as follows:

- S - Successful
- E - Expected error
- U - Unexpected error

Understanding Error Messages

These error messages are output to the stored procedure message file and are monitored through the return x_msg_count. In conjunction with the x_return_ status, this can be used to monitor the success or failure of the procedure call.

Displaying Errors in Languages Other than English

Language translation of error messages is determined by the environment variable NLS_LANGUAGE. If the message is not found in the required language, then the message is retrieved in US English.

The following is a complete list of the Cost Management API error messages. A message that is preceded with Warning is not a fatal API error. A message preceded with Error is a fatal API error.

Any uppercase word preceded by an apersand (&) is a token, or placeholder, for an actual value that populates at runtime.

Message Name	Message Code
Error: Burden Details delete failed for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_ CODE.	GMF_API_ BRDN_DEL_FAILED_ DTLS
Error: Either Allocation Id or combination of Allocation Code and Company Code is required.	GMF_API_ALLOC_DTL_REQ
Error: Allocation Method is required.	GMF_API_ALLOC_MTHD_REQ
Allocation Header. Allocation Id: &ALLOCATION_ID Allocation Code: &ALLOCATION_CODE Company: &COMPANY Item Id: &ITEM_ID Item No: &ITEM_NO Component Class Id: &CMPNT_ CLASS_ID Component Class Id: &CMPNT_ CLASS_CODE	GMF_API_ALLOCATION_HEADER
Error: Analysis Code is required.	GMF_API_ANALYSIS_CODE_REQ
Error: Balance Type is required.	GMF_API_BALANCE_TYPE_REQ
Error: Basis Account Key is required.	GMF_API_BAS_ACC_KEY_REQ
Error: YTD/PTD indicator for balance is required.	GMF_API_BAS_YTD_PTD_REQ
Error: Burden Details delete failed for Burden line Id &BURDENLINE_ID.	GMF_API_BRDN_DEL_FAILED_ID
Burden details not found for Resource &RESOURCE, Component Class Code &CMPNTCLS_CODE, Component Class Id &CMPNTCLS_ID, and Analysis Code &ALYS_ CODE.	GMF_API_BRDN_DTL_NOT_FOUND
Error: Burden Details insert failed for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_ CODE.	GMF_API_BRDN_INSERT_FAILED
Error: No Burden Details found to update for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_ CODE.	GMF_API_BRDN_NOT_FOUND_ FOR_DTL
Error: No Burden Details found to update for Burden Line Id &BURDENLINE_ID.	GMF_API_BRDN_NOT_FOUND_ FOR_ID

Message Name	Message Code
UOM Conversion failed for resource &RESOURCES from Burden UOM &BURDEN_ UM to Resource UOM &BURDEN_UM.	GMF_API_BRDN_UM_CONV_ERR
Error: Burden Details update failed for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_ CODE.	GMF_API_BRDN_UPD_FAILED_DTLS
Error: Burden Details update failed for Burden line Id &BURDENLINE_ID.	GMF_API_BRDN_UPD_FAILED_ID
Burden Details Header. Organization: &ORGN Item Id: &ITEM_ID Item No: &ITEM_NO Warehouse: &WHSE Calendar: &CALENDAR Period: &PERIOD Cost Method: &COST_ METHOD.	GMF_API_BURDEN_HEADER
Error: Burden Indicator is required.	GMF_API_BURDEN_IND_REQ
Error: Burden Quantity is required.	GMF_API_BURDEN_QTY_REQ
Error: Burden UOM is required.	GMF_API_BURDEN_UM_REQ
Error: Burden Indicator supplied is &BURDEN_ IND, but Usage of the Cost Component Class &CMPNT_CLS is not burden detail.	GMF_API_BURDEN_USAGE_IND
Error: Burden Usage is required.	GMF_API_BURDEN_USAGE_REQ
Error: Calendar Code is required.	GMF_API_CALENDAR_CODE_REQ
Error: Record cannot be marked for purge using Update API.	GMF_API_CANT_MARK_FOR_ PURGE
Error: Period &PERIOD_CODE for Calendar &CALENDAR_CODE is Closed.	GMF_API_CLOSED_PERIOD
Error: Component Cost is required.	GMF_API_CMPNT_COST_REQ
Error: Either Component Class Id or Component Class Code is required.	GMF_API_CMPNTCLS_ID_REQ
Error: Usage of Component Class &CMPNTCLS is not GL Expense Allocation.	GMF_API_CMPNTCLS_USG_NOT_ ALC
Error: Usage of Component Class &CMPNTCLS_ID is not Burden (Overhead).	GMF_API_CMPNTCLS_USG_NOT_ BRDN
Error: Cost Method Code is required.	GMF_API_COST_MTHD_CODE_REQ

Message Name	Message Code
Error: Item Cost details not found for Item Id &ITEM_ID, Warehouse &WHSE_CODE, Calendar &CALENDAR_CODE, Period &PERIOD_CODE, Cost Method &COST_MTHD, Component Class &CMPNTCLS_ID, Analysis Code &ANALYSIS_CODE and Cost Level &COST_LEVEL.	GMF_API_COST_ROWS_NOT_FOUND
DEBUG: &MSG	GMF_API_DEBUG
Error: No Burden Details found to delete for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_DEL_BRDN_NOT_FOUND_DTL
Error: No Burden Details found to delete for Burden Line Id &BURDENLINE_ID.	GMF_API_DEL_BRDN_NOT_FOUND_ID
Error: No Component Cost Details found to delete for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_DEL_IC_NOT_FOUND_DTL
Error: No Component Cost Details found to delete for Component Cost Id &CMPNTCOST_ID.	GMF_API_DEL_IC_NOT_FOUND_ID
Error: Delete Mark is required.	GMF_API_DELETE_MARK_REQ
Error: Duplicate Item Cost details for Item Id &ITEM_ID, Warehouse &WHSE_CODE, Calendar &CALENDAR_CODE, Period &PERIOD_CODE, Cost Method &COST_MTHD, Component Class &CMPNTCLS_ID, Analysis Code &ANALYSIS_CODE and Cost Level &COST_LEVEL.	GMF_API_DUPLICATE_ITEM_COST
Error: Duplicate Resource Cost for Resource Code: &RESOURCES, Orgn: &ORGN_CODE, Calendar: &CALENDAR_CODE, Period: &PERIOD_CODE and Cost Method: &COST_MTHD_CODE.	GMF_API_DUPLICATE_RSRC_COST
Error: Percentage is required.	GMF_API_FIXED_PERCENT_REQ
Error: No Updates/Deletes allowed since Period &PERIOD_CODE for Calendar &CALENDAR_CODE is Frozen.	GMF_API_FROZEN_PERIOD

Message Name	Message Code
Error: Cannot insert new component details since Item &ITEM is frozen for Whse &WHSE, Calendar &CALENDAR, Period &PERIOD and Cost Method &COST_MTHD.	GMF_API_IC_CANNT_INSERT_CMPTS
Error: Component Cost delete failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_DEL_FAILED_DTLS
Error: Component Cost delete failed for Component Cost Id &CMPNTCOST_ID.	GMF_API_IC_DEL_FAILED_ID
Error: Lower Level Component Cost insert failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_LWRLVL_INS_FAILED
Error: This Level Component Cost insert failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_THISLVL_INS_FAILED
Error: Component Cost update failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_UPD_FAILED_DTLS
Error: Component Cost update failed for Component Cost Id &CMPNTCOST_ID.	GMF_API_IC_UPD_FAILED_ID
Warning: Allocation Code &ALLOC_CODE and Company Code &CO_CODE values are ignored since allocation Id is supplied.	GMF_API_IGNORE_ALLOC_CODE
Warning: Basis Amount fields are ignored since fixed percent is used for Allocation (Allocation Method is Fixed Percent).	GMF_API_IGNORE_BASIS
Warning: BurdenLine Id &BURDENLINE_ID is supplied and will be used to Update/Delete the record. Unique key combination of Organization, Item, Warehouse, Calendar, Period, Cost Method, Resource, Component Class and Analysis Code will be ignored.	GMF_API_IGNORE_BRDN_UNIQUE_KEY
Warning: Component Class Code &CMPNTCLS_CODE is ignored since Component Class Id is supplied.	GMF_API_IGNORE_CMPNTCLS_CODE
Warning: Fixed Percent &FIXED_PERCENT value is ignored since Alloc Percent Depends on Basis Amount (Allocation Method is Basis Type).	GMF_API_IGNORE_FIXED_PERCENT

Message Name	Message Code
Warning: Component Cost Id &CMPNTCOST_ID is supplied and will be used to Update/Delete the record. Unique key combination of Item, Warehouse, Calendar, Period, Cost Method, Component Class, Analysis Code and Cost Level will be ignored.	GMF_API_IGNORE_IC_UNIQUE_KEY
Warning: Item No &ITEM_NO is ignored since Item Id is supplied.	GMF_API_IGNORE_ITEM_NO
Warning: Lower level details cannot be updated or deleted hence ignored.	GMF_API_IGNORE_LOWER_LEVEL
Error: Invalid Accouting No in the Basis Account Key &BAS_ACC_KEY.	GMF_API_INVALID_ACCT_NO
Error: Invalid Accouting Unit in the Basis Account Key &BAS_ACC_KEY.	GMF_API_INVALID_ACCTG_UNIT
Error: Invalid combination of Allocation Code &ALLOC_CODE and Company &CO_CODE.	GMF_API_INVALID_ALLOC_CODE
Error: Allocation Method &ALLOC_METHOD not consistent with the existing method.	GMF_API_INVALID_ALLOC_DEF
Error: Invalid Allocation Id: &ALLOC_ID.	GMF_API_INVALID_ALLOC_ID
Error: Invalid Allocation Method: &ALLOC_METHOD.	GMF_API_INVALID_ALLOC_MTHD
Error: Invalid Analysis Code: &ANALYSIS_CODE.	GMF_API_INVALID_ANALYSIS_CODE
Error: Invalid Balance Type: &BALANCE_TYPE.	GMF_API_INVALID_BALANCE_TYPE
Error: Invalid Basis amount Indicator: &BAS_YTD_PTD.	GMF_API_INVALID_BAS_YTD_PTD
Invalid burden line id &BURDENLINE_ID passed.	GMF_API_INVALID_BRDN_LINE_ID
Error: Invalid Burden Indicator: &BURDEN_IND.	GMF_API_INVALID_BURDEN_IND
Error: Invalid Burden Quantity: &BURDEN_QTY (Should be between 0.000000001 and 999999999D999999999).	GMF_API_INVALID_BURDEN_QTY
Error: Invalid Burden UOM: &BURDEN_UM.	GMF_API_INVALID_BURDEN_UM

Message Name	Message Code
Error: Invalid Burden Usage: &BURDEN_ USAGE (Should be between 0.000000001 and 999999999D999999999).	GMF_API_INVALID_BURDEN_ USAGE
Error: Invalid Calendar Code: &CALENDAR_ CODE.	GMF_API_INVALID_CALENDAR_ CODE
Error: Component cost &CMPNT_COST should not be greater than 999,999,999.999999999.	GMF_API_INVALID_CMPNT_COST
Error: Invalid Component Class Code: &CMPNTCLS_CODE.	GMF_API_INVALID_CMPNTCLS_ CODE
Error: Invalid Component Class Id: &CMPNTCLS_ID.	GMF_API_INVALID_CMPNTCLS_ID
Error: Invalid Cost Method Code: &COST_ MTHD_CODE.	GMF_API_INVALID_COST_MTHD_ CODE
Error: Invalid Delete Mark: &DELETE_MARK.	GMF_API_INVALID_DELETE_MARK
Error: Invalid Percentage: &FIXED_PERCENT (Should be 0 to 100).	GMF_API_INVALID_FIXED_ PERCENT
Error: Invalid Item Id: &ITEM_ID.	GMF_API_INVALID_ITEM_ID
Error: Invalid Item No: &ITEM_NO.	GMF_API_INVALID_ITEM_NO
Error: Invalid Item Quantity: &ITEM_QTY (Should be between 0.000000001 and 999999999D999999999).	GMF_API_INVALID_ITEM_QTY
Error: Invalid Item UOM: &ITEM_UM.	GMF_API_INVALID_ITEM_UM
Error : Invalid Lot Id : &LOT_ID for Item Id : &ITEM_ID	GMF_API_INVALID_LOT_ID
Error : Invalid Lot No : &LOT_NO for Sublot No : &SUBLOT_NO Item Id : &ITEM_ID	GMF_API_INVALID_LOT_NO
Error: Invalid Nominal Cost: &NOMINAL_ COST (Should be between 0 and 999999999D999999999).	GMF_API_INVALID_NOMINAL_ COST
Error: Invalid Organization Code: &ORGN_ CODE.	GMF_API_INVALID_ORGN_CODE
Error: Invalid Period Code: &PERIOD_CODE for Calendar &CALENDAR_CODE.	GMF_API_INVALID_PERIOD_CODE
Error: Invalid Resource Code: &RESOURCES.	GMF_API_INVALID_RESOURCES

Message Name	Message Code
Error: For Actual Cost method, Raw Material Cost Calculation Type should be 1-PMAC, 2-PWAC, 3-PPAC, 4-LSTT or 5 -LSTI.	GMF_API_INVALID_RMCALC_TYPE
Error: Invalid Usage Unit of Measure: &USAGE_UM.	GMF_API_INVALID_USAGE_UM
Error: Invalid User name: &USER_NAME.	GMF_API_INVALID_USER_NAME
Error: Invalid Warehouse Code: &WHSE_CODE.	GMF_API_INVALID_WHSE_CODE
Error: Either Item Id or Item No is required.	GMF_API_ITEM_ID_REQ
Error: Item Quantity is required.	GMF_API_ITEM_QTY_REQ
Error: UOM conversion error for Item &ITEM_ID. Cannot convert quantity from current Item UOM &ITEM_UM to actual Item UOM &ITEM_ACT_UM.	GMF_API_ITEM_UM_CONV_ERR
Error: Item UOM is required.	GMF_API_ITEM_UM_REQ
Item Cost Header. Calendar: &CALENDAR Period: &PERIOD Cost Method: &COST_METHOD Warehouse: &WHSE Item Id: &ITEM_ID Item No: &ITEM_NO.	GMF_API_ITEMCOST_HEADER
Error : Invalid Adjustment Cost	GMF_API_LCA_ADJ_COST
Error : Invalid Adjustment Date	GMF_API_LCA_ADJ_DATE
Error : Invalid Adjustment Id	GMF_API_LCA_ADJUSTMENT_ID
Item : &ITEM Company : &COMPANY Cost Method : &COST_METHOD Warehouse : &WAREHOUSE Lot : &LOT Adjustment Date : &ADJUSTMENT_DATE	GMF_API_LCA_DEL_FAILED
Error : Lot Cost Adjustment Detail Delete failed for Adjustment Details ID : &ADJUSTMENT_DTL_ID	GMF_API_LCA_DTL_ID_DEL_FAILED
Error : Lot Cost Adjustment Detail Update failed for Adjustment Details ID : &ADJUSTMENT_DTL_ID	GMF_API_LCA_DTL_ID_UPD_FAILED
Error : Lot Cost Adjustment Detail Insert failed for Component Class : &COMPONENT_CLASS Analysis Code : &ANALYSIS_CODE	GMF_API_LCA_DTL_INS_FAILED

Message Name	Message Code
Error : Lot Cost Adjustment Header Insert failed for Item : &ITEM Company : &COMPANY Cost Method : &COST_METHOD Warehouse : &WAREHOUSE Lot : &LOT Adjustment Date : &ADJUSTMENT_DATE	GMF_API_LCA_INS_FAILED
Error : Invalid Reason Code : &REASON_CODE	GMF_API_LCA_REASON_CODE
Error: Line Number is required.	GMF_API_LINE_NO_REQ
Error : Either Lot Id or Lot No is required.	GMF_API_LOT_ID_REQ
Message: No rows deleted.	GMF_API_NO_ROWS_DEL
Error: No rows found to update/delete for Allocation Id &ALLOC_ID and Line No &LINE_NO.	GMF_API_NO_ROWS_FOUND
Message: No rows inserted.	GMF_API_NO_ROWS_INS
Message: No rows updated.	GMF_API_NO_ROWS_UPD
Error: Nominal Cost is required.	GMF_API_NOMINAL_COST_REQ
Error: Organization Code is required.	GMF_API_ORGN_CODE_REQ
Error: Period Code is required.	GMF_API_PERIOD_CODE_REQ
Resource Cost Header. Resource: &RESOURCES Organization: &ORGN Calendar: &CALENDAR Period: &PERIOD Cost Method: &COST_MTHD	GMF_API_RESOURCE_HEADER
Error: Resource Code is required.	GMF_API_RESOURCES_REQ
&NUM_ROWS row(s) deleted.	GMF_API_ROWS_DEL
&NUM_ROWS row(s) inserted.	GMF_API_ROWS_INS
&NUM_ROWS row(s) updated.	GMF_API_ROWS_UPD
Error: No rows found to update/delete for Resource &RESOURCES, Orgn &ORGN_CODE, Calendar &CALENDAR_CODE, Period &PERIOD_CODE and Cost Method &COST_MTHD_CODE.	GMF_API_RSRC_NO_ROWS_FOUND
Error : Invalid Text Code : &TEXT_CODE	GMF_API_TEXT_CODE
Warning: Total Fixed Percentage for Allocation Id &ALLOC_ID is not equal to 100.	GMF_API_TOTAL_PCT_NOTHUNDRED

Message Name	Message Code
Error: Burden UOM &BURDEN_UM must be of the same type as the resource UOM &RESOURCE_UM.	GMF_API_UOM_SAMETYPE_REQ
Error: No Component Cost Details found to update for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_UPD_IC_NOT_FOUND_DTL
Error: No Component Cost Details found to update for Component Cost Id &CMPNTCOST_ID.	GMF_API_UPD_IC_NOT_FOUND_ID
Error: Usage Unit of Measure is required.	GMF_API_USAGE_UM_REQ
Error : Usage UOM &USAGE_UM must be of the same type as the resource UOM &RESOURCE_UM.	GMF_API_USAGE_UOM_SAMETYPE_REQ
User Name is required.	GMF_API_USER_NAME_REQ
Error: Warehouse Code is required.	GMF_API_WHSE_CODE_REQ
Adjustment date should be before today's date (&SYSDATE). Please correct	GMF_LCA_ADJ_DATE
Adjustment date should be between last cost transaction date(&REFDATE) and today's date(&SYSDATE). Please correct.	GMF_LCA_ADJ_DATE_RANGE

Glossary

Application Programming Interface (API)

A documented, supported method for communicating within or between modules.

Business Object

An independent item of significance in the business world. An example of a Business Object is a sales order.

Business Process API

An API that performs a transaction for the calling module. For example, to hire an employee, enter an order, or cost a material movement transaction.

Entity

An item of significance in the business world, that has no meaning without reference to a Business Object. An example of an Entity is a Sales Order Header. A Sales Order Header is an entity of the business object Sales Order.

Group API

An API intended for use by other authorized Oracle Applications modules.

Module

A module is a collection of one or more business objects and the associated transactions. A module publishes APIs for other modules and accesses other modules through their published APIs. An example of a module is Oracle Inventory.

Public API

An API intended for use by all applications; contrast to *Private API.*

Private API

An API intended for use by the owning module only; contrast to *published API.*

Index

I

L

M

P

R

S

T

U

V

X