Contents

Send Us Your Comments ...................................................................................................................................................... vii

Preface ...................................................................................................................................................................................... ix

Audience for This Guide ......................................................................................................................................................... ix

How To Use This Guide ......................................................................................................................................................... ix

Documentation Accessibility ..................................................................................................................................................... x

Other Information Sources ................................................................................................................................................... x

Online Documentation ............................................................................................................................................................ x

Related User’s Guides ............................................................................................................................................................ xi

Guides Related to All Products ................................................................................................................................................ xi

User Guides Related to This Product ....................................................................................................................................... xi

Installation and System Administration ......................................................................................................................................... xvi

Other Implementation Documentation ........................................................................................................................................ xvi

Training and Support .............................................................................................................................................................. xix

Do Not Use Database Tools to Modify Oracle Applications Data .............................................................................................. xx

About Oracle ............................................................................................................................................................................. xxi

Your Feedback ........................................................................................................................................................................... xxi

1 Cost Management APIs

Introducing the Cost Management APIs ................................................................................................................................ 1-2

Basic Business Needs ................................................................................................................................................................. 1-2

Major Features ........................................................................................................................................................................... 1-3

Item Cost APIs ........................................................................................................................................................................... 1-4

Resource Cost APIs .................................................................................................................................................................... 1-4
2 Cost Management API Usage

Calling the API Interface Code ................................................................. 2-1
Calling the API Code - Example ................................................................. 2-2
Wrapper logic .............................................................................................. 2-2
API Hints...................................................................................................... 2-4
  Item Cost Example....................................................................................... 2-5

3 Technical Overview

Item Cost ....................................................................................................... 3-1
  Structure for Item Cost Public APIs......................................................... 3-1
Resource Cost.............................................................................................. 3-3
  Structure for Resource Cost Public APIs.............................................. 3-3
Allocation Definition .................................................................................. 3-4
  Structure for Allocation Definition Public APIs................................. 3-4
Burden Details............................................................................................. 3-5
  Structure for Burden Details Public APIs.............................................. 3-5
Standard Parameters.................................................................................. 3-6

4 Business Objects

Item Cost....................................................................................................... 4-1
  Cost Header Entity ................................................................................. 4-2
  This Level Cost Detail Entity .............................................................. 4-3
  Lower Level Cost Detail Entity ........................................................... 4-4
  Structure and Logic .............................................................................. 4-5
Resource Cost............................................................................................. 4-12
  Resource Cost Entity ......................................................................... 4-12
  Structure and Logic ........................................................................... 4-13
Allocation Definition ................................................................. 4-17
  Allocation Definition Entity .................................................. 4-17
  Structure and Logic ............................................................ 4-18
Burden Details ............................................................................. 4-22
  Header Entity ....................................................................... 4-22
  Detail Entity ......................................................................... 4-23
  Structure and Logic ............................................................ 4-24

A Messages and Errors
  Handling Messages ............................................................... A-1
  Interpreting Error Conditions .............................................. A-4
  Understanding Error Messages .......................................... A-4

Index
Send Us Your Comments

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

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

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

- FAX: 650-506-7200  Attn: Oracle Process Manufacturing
- Postal service:
  Oracle Corporation
  Oracle Process Manufacturing
  500 Oracle Parkway
  Redwood City, CA 94065
  U.S.A.
- Electronic mail message to appsdoc@us.oracle.com

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

Audience for This Guide


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 Cost Management APIs, Oracle suggests you attend one or more of the Oracle Process Manufacturing Cost Management APIs training classes available through Oracle University.
- 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

This guide contains the information you need to understand and use Oracle Process Manufacturing.

- 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.
- 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 Corporation 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.

Other Information Sources

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

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

Online Documentation

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

- Online Help - The new features section in the HTML help describes new features in 11i. This information is updated for each new release of Oracle Process Manufacturing Cost Management APIs. The new features section also includes information about any features that were not yet available when this guide was printed. For example, if your administrator has installed software from a mini-packs an upgrade, this document describes the new features. Online help patches are available on MetaLink.
- **11i Features Matrix** - This document lists new features available by patch and identifies any associated new documentation. The new features matrix document is available on MetaLink.

- **Readme File** - Refer to the readme file for patches that you have installed to learn about new documentation or documentation patches that you can download.

### Related User's Guides

*Oracle Process Manufacturing* shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user’s guides when you set up and use *Oracle Process Manufacturing Cost Management APIs.*

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) available with this release of *Oracle Process Manufacturing Cost Management APIs* (and any other Oracle Applications products). This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

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

### User 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.

Integration with Advanced Planning and Scheduling User’s Guide
Oracle Process Manufacturing and Oracle Advanced Planning and Scheduling (APS) combine to provide an integrated solution for process manufacturers that can help increase planning efficiency. The integration 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 Oracle Process Manufacturing Integration with Advanced Planning and Scheduling 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.

Regulatory Management User’s Guide
The Oracle Process Manufacturing Regulatory Management application generates the Material Safety Data Sheets (MSDSs) required by authorities to accompany hazardous materials during shipping. You can create MSDSs from OPM Formula
Management with Regulatory or Production effectivities. The Oracle Process Manufacturing Regulatory Management User’s Guide describes how to setup 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**


**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 11i. 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 11i, much of the installation process is handled using Oracle Rapid Install, which minimizes the time to install Oracle Applications, the Oracle8 technology stack, and the Oracle8i Server technology stack by automating many of the required steps. This guide contains instructions for using Oracle Rapid Install and lists the tasks you need to perform to finish your installation. You should use this guide in conjunction with individual product user’s guides and implementation guides.

**Upgrading Oracle Applications**
Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11i. 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 11i. You cannot upgrade to Release 11i directly from releases prior to 10.7.

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

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

**Oracle Alert User’s Guide**
This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.
Oracle Applications Developer's Guide
This guide contains the coding standards followed by the Oracle Applications
development staff. It describes the Oracle Application Object Library components
needed to implement the Oracle Applications user interface described in the Oracle
Applications User Interface Standards for Forms-Based Products. It also provides
information to help you build your custom Oracle Forms Developer 6i forms so that
they 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 11i. It includes new features, enhancements, and
changes made to database objects, profile options, and seed data for this interval.

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

Multiple Organizations in Oracle Applications
This guide describes how to set up and use Oracle Process Manufacturing with Oracle
Applications' Multiple Organization support feature, so you can define and support
different organization structures when running a single installation of Oracle Process
Manufacturing.

Oracle Workflow Guide
This guide explains how to define new workflow business processes as well as
customize existing Oracle Applications-embedded workflow processes. You also use
this guide to complete the setup steps necessary for any Oracle Applications product that includes workflow-enabled processes.

**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 manual also provides information on creating custom reports on flexfields data.

**Oracle eTechnical Reference Manuals**

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

**Oracle Manufacturing APIs and Open Interfaces Manual**

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

**Oracle Order Management Suite APIs and Open Interfaces Manual**

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

**Oracle Applications Message Reference Manual**

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

**Training and Support**

**Training**

Oracle offers a complete set of training courses to help you and your staff master Oracle Process Manufacturing Cost Management APIs 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 Cost Management APIs 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 Oracle8i 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 Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 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 Cost Management APIs and this user’s guide.

Oracle values your comments and feedback. At the end of this guide is a Reader’s Comment Form you can use to explain what you like or dislike about Oracle Process Manufacturing Cost Management APIs or this user’s 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.
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 allow 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 would 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 module-specific objects that allow 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 module.

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 will also be responsible for connecting to a database before calling an API function, and disconnecting from the database upon return. You may also choose to write log files before calling and after returning from a function. If there is a problem during execution of a call, the APIs return one of the following status codes:

- S for success
- E for error
- U unknown/unexpected status.

Basic Business Needs

These APIs allow users to 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 screen, through which users can manipulate cost information in the Oracle Process Manufacturing (OPM) application. The usage of this API will be multifold:

- 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.
Major Features

- 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 will also serve 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 should have:

**Code Re-Use**
You should be able to maximize code reuse from all application development tools, including PL/SQL, Oracle Forms and Oracle Reports.

**Ease of Integration**
You should be able to integrate API’s into other applications and enabling technology, such as Oracle Workflow Server, Oracle Internet Commerce & Oracle WebSystem, and Oracle EDI Gateway.

**Insulation from Changes**
You should be able to encapsulate the structure of schema to prevent changing schema structures from affecting other modules.

**Consistent Behavior**
You should be able to hide Object logic specific to a module from other modules, and to ensure that this logic is correctly invoked by other modules and customers.

**Robust Validation**
You should be able to fully validate all incoming information into Oracle Applications.

---

**Major Features**

In order to support requirements mentioned in “Basic Business Needs” section, new APIs will be developed supporting following functionality on business object - **Item Cost**, **Resource Cost**, **Allocation Definition**, and **Burden Details**.
Major Features

Item Cost APIs

Insert Item Cost
This API will be used to insert new Item Cost record in OPM. It will help new customers of OPM load Item Cost information into OPM from external systems.

Update Item Cost
This API will be used to update Item Cost record for an existing Item. Using this API, user can update Item Cost in OPM.

Delete Item Cost
This API will be used to delete Item Cost record in OPM. Using this API, user can delete Item Cost record.

Retrieve Item Cost
This API will be used to retrieve Item Cost information from OPM. Using this API, user can retrieve Item Cost information from OPM to be used by external systems or for any other purpose.

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

Resource Cost APIs

Insert Resource Cost
This API will be used to insert new Resource Cost record in OPM. It will help new customers of OPM load Resource Cost information into OPM from external systems.

Update Resource Cost
This API will be used to update Resource Cost record for an existing Resource. Using this API, user can update Resource Cost in OPM.
Delete Resource Cost
This API will be used to delete Resource Cost record in OPM. Using this API, user can delete Resource Cost record.

Retrieve Resource Cost
This API will be used to retrieve Resource Cost information from OPM. Using this API, user 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 will be used to insert new Allocation Definition record in OPM. It will help customers of OPM load allocation definition information into OPM from external systems.

Update Allocation Definition
This API will be used to update an existing Allocation Definition record. Using this API, user can update allocation definition in OPM.

Delete Allocation Definition
This API will be used to delete Allocation Definition record in OPM. Using this API, user can delete allocation definition record.

Burden Details APIs

Insert Burden Details
This API will be used to insert new Burden Details in OPM. It will help new customers of OPM load Burden Details information into OPM from external systems.

Update Burden Details
This API will be used to update Burden Details for an existing item and resource. Using this API, user can update Burden Details in OPM.
**Delete Burden Details**
This API will be used to delete Burden Details record in OPM. Using this API, user can delete Burden Details record.

**Retrieve Burden Details**
This API will be used to retrieve Burden Details information from OPM. Using this API, user can retrieve Burden Details information from OPM to be used by external systems or for any other purpose.

**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 will provide objects and libraries needed to use to the APIs and the documentation for their use.
- Oracle will ensure that the APIs function as designed.
- Oracle will 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 manual 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.

<table>
<thead>
<tr>
<th>Package Name</th>
<th>File Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMF_ItemCost_PUB</td>
<td>GMFPCSTS.pls</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>GMFPCSTB.pls</td>
<td></td>
</tr>
<tr>
<td>GMF_ResourceCost_PUB</td>
<td>GMFPRESS.pls</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>GMFPCSTB.pls</td>
<td></td>
</tr>
<tr>
<td>GMF_AllocationDefinition_PUB</td>
<td>GMFPALCS.pls</td>
<td>Public Cost Management package that the user defined function calls. The business API can be used for inserting, updating, or deleting an allocation definition.</td>
</tr>
<tr>
<td></td>
<td>GMFPCSTB.pls</td>
<td></td>
</tr>
<tr>
<td>GMF_BurdenDetails_PUB</td>
<td>GMFPBRDS.pls</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>GMFPBRDB.pls</td>
<td></td>
</tr>
</tbody>
</table>
The Cost Management APIs are currently written in PL/SQL. To make use of these API’s, you must code your interface or wrapper. Your program will also be 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 will return 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 will probably want to 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 i.e., 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 should correspond 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 in case of 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:
   a. 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 (i.e., 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:

```sql
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:

```java
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:

```java
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 );
```
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   --|
FUNCTION Create_Item_Cost
  ( p_dir          IN VARCHAR2
  , p_input_file   IN VARCHAR2
  , p_output_file  IN VARCHAR2
  , p_delimiter    IN VARCHAR2
  )
  RETURN VARCHAR2
IS
  /* Local variables */
  l_status           VARCHAR2(11);
  l_return_status    VARCHAR2(11) := FND_API.G_RET_STS_SUCCESS;
  l_count            NUMBER(10) ;
  l_record_count     NUMBER(10) :=0;
  l_loop_cnt         NUMBER(10) :=0;
  l_dummy_cnt        NUMBER(10) :=0;
  l_data             VARCHAR2(1000);
  l_status           VARCHAR2(11);
BEGIN
/* Enable The Buffer */
DBMS_OUTPUT.ENABLE(1000000);

/* Obtain The SessionId To Append To wrapper File Name. */

l_session_id := USERENV('sessionid');

l_log_name := CONCAT(l_log_name, l_session_id);

/* Directory is now the same same as for the out file */
l_log_dir := p_dir;

/* Open The Wrapper File For Output And The Input File for Input. */

l_log_handle := UTL_FILE.FOPEN(l_log_dir, l_log_name, 'w');
l_infile_handle := UTL_FILE.FOPEN(l_p_dir, l_input_file, 'r');

/* Loop thru flat file and call Item Cost API */
DBMS_OUTPUT.PUT_LINE('Process Started at ' || to_char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
DBMS_OUTPUT.PUT_LINE('Input Directory  ' || l_p_dir);
DBMS_OUTPUT.PUT_LINE('Input File       ' || l_input_file);
DBMS_OUTPUT.PUT_LINE('Delimiter        ' || l_delimiter);
DBMS_OUTPUT.PUT_LINE('Output File      ' || l_output_file);

DBMS_OUTPUT.PUT_LINE('Start Processing');
UTL_FILE.PUT_LINE(l_log_handle, 'Process Started at ' || to_char(SYSDATE,'DD-MON-YY HH24:MI:SS'));

UTL_FILE.PUT_LINE(l_log_handle, 'Input Directory  ' || l_p_dir);
UTL_FILE.PUT_LINE(l_log_handle, 'Input File       ' || l_input_file);
UTL_FILE.PUT_LINE(l_log_handle, 'Record Type      ' || l_delimiter);
UTL_FILE.PUT_LINE(l_log_handle, 'Output File      ' || l_output_file);

l_outfile_handle := UTL_FILE.FOPEN(l_p_dir, l_output_file, 'w');

/* Get the first record from the file */
BEGIN
  UTL_FILE.GET_LINE(l_infile_handle, l_line);
l_record_count := l_record_count+1;
l_type := Get_Field(l_line, l_delimiter, 1); -- 10 : header rec, 20 : this level, 30 : lower level
END;

EXCEPTION
WHEN NO_DATA_FOUND THEN
  raise;
END;

/* Populate appropriate pl/sql record or table based on the type of record */
LOOP
  BEGIN
    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) ;

2-10  Product Title/BookTitle as a Variable
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||')      =
API Hints

Cost Management API Usage

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

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

2-14  Product Title/BookTitle as a Variable
:= ' || 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  IF l_count > 0 THEN
343
344    l_loop_cnt :=1;
345    LOOP
346      FND_MSG_GET.Get(
347        p_msg_index     => l_loop_cnt,
348        p_data          => l_data,
349        p_encoded       => FND_API.G_FALSE,
350        p_msg_index_out => l_dummy_cnt);
351
352      --DBMS_OUTPUT.PUT_LINE(l_data );
353      UTL_FILE.PUT_LINE(l_log_handle, l_data);
354
355      /* Update error status */
356      IF (l_status = 'U') THEN
357        l_return_status  :=l_status;
358      ELSIF (l_status = 'E' and l_return_status <> 'U') THEN
359        l_return_status  :=l_status;
360      ELSE
361        l_return_status :=l_status;
362      END IF;
363
364      l_loop_cnt  := l_loop_cnt + 1;
365      IF l_loop_cnt > l_count THEN
366        EXIT;
367      END IF;
368
369    END LOOP; -- msg stack loop
370
371    l_count := 0 ;
372
373  END IF;-- if count of msg stack > 0
374
375  DBMS_OUTPUT.PUT_LINE('### 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, 'CmpntClsId : ' || l_costcmpnt_ids(i).cost_cmpntcls_id || ' Analysis Code : ' || l_costcmpnt_ids(i).cost_analysis_code || ' Cost Level : ' || l_costcmpnt_ids(i).cost_level || ' 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  WHEN UTL_FILE.READ_ERROR THEN
423      /* DBMS_OUTPUT.PUT_LINE('Invalid Read Error ' || l_global_file); */
424      UTL_FILE.FCLOSE_ALL;
425      RETURN l_return_status;
426
427  WHEN UTL_FILE.INTERNAL_ERROR THEN
428      /* DBMS_OUTPUT.PUT_LINE('Internal Error'); */
429      UTL_FILE.FCLOSE_ALL;
430      RETURN l_return_status;
431
432  WHEN OTHERS THEN
433      /* DBMS_OUTPUT.PUT_LINE('Other Error'); */
434      UTL_FILE.PUT_LINE(l_outfile_handle, 'Error : ' || to_char(SQLCODE) || ' ' || SQLERRM);
435      UTL_FILE.PUT_LINE(l_log_handle, 'Error : ' || to_char(SQLCODE) || ' ' || SQLERRM);
436      UTL_FILE.PUT_LINE(l_log_handle, 'Process Completed at ' || to_char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
437      UTL_FILE.FCLOSE_ALL;
438      l_return_status := 'U' ;
439      RETURN l_return_status;
440
441  END Create_Item_Cost;
442
443
444
445  --+==========================================================================+
446  --| FUNCTION NAME
447  --| Get_Field
448  --| TYPE
449  --| Public
450  --| USAGE
451  --| Get value of field n from a delimited line of ASCII data
452  --|
453  --|
454  --|
```
-- DESCRIPTION
This utility function will return the value of a field from
a delimited line of ASCII text

-- PARAMETERS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>p_line</td>
<td>IN VARCHAR2</td>
<td>- line of data</td>
</tr>
<tr>
<td>p_delimiter</td>
<td>IN VARCHAR2</td>
<td>- Delimiter character</td>
</tr>
<tr>
<td>p_field_no</td>
<td>IN NUMBER</td>
<td>- Field occurrence to be returned</td>
</tr>
</tbody>
</table>

-- RETURNS

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
</tr>
<tr>
<td>VARCHAR2</td>
</tr>
</tbody>
</table>

-- HISTORY

-- Api end of comments

FUNCTION Get_Field
( p_line IN VARCHAR2,
  p_delimiter IN VARCHAR2,
  p_field_no IN NUMBER
) RETURN VARCHAR2 IS

/* Local variables */

l_start NUMBER :=0;
483  l_end NUMBER :=0;
484
485  BEGIN
486
487  /* Determine start position */
488  IF p_field_no = 1
489  THEN
490    l_start :=0;
491  ELSE
492    l_start :=INSTR(p_line,p_delimiter,1,(p_field_no - 1));
493    IF l_start = 0
494      RETURN NULL;
495    END IF;
496  END IF;
497  END IF;
498
499  /* Determine end position */
500  l_end :=INSTR(p_line,p_delimiter,1,p_field_no);
501  IF l_end = 0
502    l_end := LENGTH(p_line) + 1;
503  END IF;
504
505  /* Extract the field data */
506  IF (l_end - l_start) = 1
507    RETURN NULL;
508  THEN
509    RETURN SUBSTR(p_line,(l_start + 1),((l_end - l_start) - 1));
510  END IF;
511
512  EXCEPTION
513    WHEN OTHERS
514    THEN
515    RETURN NULL;
516  END IF;
517
518  END Get_Field;
519
520
521  --==========================================================================
522  --| FUNCTION NAME
523    --| Get_Substring
524    --|
TYPE

Public

USAGE

Get value of Sub-string from formatted ASCII data file record

DESCRIPTION

This utility function will return the value of a passed sub-string

of a formatted ASCII data file record

PARAMETERS

p_substring        IN VARCHAR2         - substring data

RETURNS

VARCHAR2                               - Value of field

HISTORY

FUNCTION Get_Substring

FUNCTION Get_Substring

(p_substring    IN VARCHAR2

RETURN VARCHAR2

2-20  Product Title/BookTitle as a Variable
550  IS
551  /* Local variables */
552  l_string_value VARCHAR2(200) := ' ';  
553  BEGIN
554  /* Determine start position */
555  l_string_value := NVL(RTRIM(LTRIM(p_substring)),' ');  
556  RETURN l_string_value;
557  EXCEPTION
558  WHEN OTHERS
559  THEN
560  RETURN ' ';  
561  END Get_Substring;
Product Title/BookTitle as a Variable
Item Cost

Each function on 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 will perform all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. In case of any validation errors the API stops the processing and returns an error status without finishing the remaining detail records. During the insert (i.e., after the validations to all detail records) if insert fails for any reason, then all the detail records for the item in process will not be inserted. After finishing validations on input data, public API will perform the required function by calling necessary routines.

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

Structure for Item Cost Public APIs

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Specification File</td>
<td>GMFPCSTS.pls</td>
</tr>
<tr>
<td>Package Body File</td>
<td>GMFPCSTB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>GMF_ItemCost_PUB</td>
</tr>
<tr>
<td>Procedure - Create Item Cost</td>
<td>Create_Item_Cost</td>
</tr>
<tr>
<td>Procedure - Update Item Cost</td>
<td>Update_Item_Cost</td>
</tr>
<tr>
<td>Procedure - Delete Item Cost</td>
<td>Delete_Item_Cost</td>
</tr>
<tr>
<td><strong>Object Type</strong></td>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Procedure - Get Item Cost</td>
<td>Get_Item_Cost</td>
</tr>
</tbody>
</table>
Resource Cost

Each function on 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 will perform all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. In case of any validation errors that particular row will be skipped and the process will continue with the next Resource Cost record. During the insert (i.e., after the validations to all detail records) if insert fails for any reason, then Resource Cost record in process will not be inserted and process will continue with the next Resource Cost record. After finishing validations on input data, public API will perform the required function by calling necessary routines.

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

**Structure for Resource Cost Public APIs**

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Specification File</td>
<td>GMFPRESS.pls</td>
</tr>
<tr>
<td>Package Body File</td>
<td>GMFPRESB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>GMF_ResourceCost_PUB</td>
</tr>
<tr>
<td>Procedure - Create Resource Cost</td>
<td>Create_Resource_Cost</td>
</tr>
<tr>
<td>Procedure - Update Resource Cost</td>
<td>Update_Resource_Cost</td>
</tr>
<tr>
<td>Procedure - Delete Resource Cost</td>
<td>Delete_Resource_Cost</td>
</tr>
<tr>
<td>Procedure - Get Resource Cost</td>
<td>Get_Resource_Cost</td>
</tr>
</tbody>
</table>
Allocation Definition

Each function on 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 will perform all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. In case of any validation errors that particular row will be skipped and the process will continue with the next Allocation Definition record. During the insert (i.e., after the validations to all detail records) if insert fails for any reason, then Allocation Definition record in process will not be inserted and process will continue with the next Allocation Definition record. After finishing validations on input data, public API will perform the required function by calling necessary routines.

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

**Structure for Allocation Definition Public APIs**

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Specification File</td>
<td>GMFPALCS.pls</td>
</tr>
<tr>
<td>Package Body File</td>
<td>GMFPALCB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>GMF_AllocationDefinition_PUB</td>
</tr>
<tr>
<td>Procedure - Create Allocation Definition</td>
<td>Create_Allocation_Definition</td>
</tr>
<tr>
<td>Procedure - Update Allocation Definition</td>
<td>Update_Allocation_Definition</td>
</tr>
<tr>
<td>Procedure - Delete Allocation Definition</td>
<td>Delete_Allocation_Definition</td>
</tr>
</tbody>
</table>
Burden Details

Each function on 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 will perform all validations necessary on input data supplied in order to prevent the flow of invalid data into OPM. In case of any validation errors the API stops the processing and returns an error status without finishing the remaining detail records. During the insert (i.e., after the validations to all detail records) if insert fails for any reason, then all the Burden Detail records for the item in process will not be inserted. After finishing validations on input data, public API will perform the required function by calling necessary routines.

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

Structure for Burden Details Public APIs

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Specification File</td>
<td>GMFPBRDS.pls</td>
</tr>
<tr>
<td>Package Body File</td>
<td>GMFPBRDB.pls</td>
</tr>
<tr>
<td>Package</td>
<td>GMF_BurdenDetails_PUB</td>
</tr>
<tr>
<td>Procedure - Create Burden Details</td>
<td>Create_Burden_Details</td>
</tr>
<tr>
<td>Procedure - Update Burden Details</td>
<td>Update_Burden_Details</td>
</tr>
<tr>
<td>Procedure - Delete Burden Details</td>
<td>Delete_Burden_Details</td>
</tr>
</tbody>
</table>
Standard Parameters

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

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

On the other hand, 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 will be 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>IN/OUT</th>
<th>Required</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_api_version</td>
<td>varchar2</td>
<td>IN</td>
<td>Y</td>
<td>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.</td>
</tr>
<tr>
<td>p_init_msg_list</td>
<td>varchar2</td>
<td>IN</td>
<td>N</td>
<td>Used to specify whether the message list should be initialized on entry to the API. It is an optional parameter, and if not supplied will default to FND/APLG_FALSE which means that the API will not initialize the message list.</td>
</tr>
</tbody>
</table>
**Value-ID Conversion**

IDs are usually used to represent primary and foreign entity keys, they are also used for internal processing of attributes. They are usually not meaningful to users and thus are usually hidden. Besides IDs, attributes may have values that represent them. Those values are usually meaningful to users and thus are used for display purposes. In general, API operates 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, i.e., 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 will be generated to inform the API caller that some of his/her input has been ignored.
This message isn’t an error message. The API will then continue with its regular processing.

Each value has to resolve into one and only one ID. Failure to resolve a value into an ID (for any reason) is an error and will be associated with an error message. The API will then abort processing and return with a return status of error.
Item Cost

Item Cost specifies cost of a given Item. To understand how it is constructed, let’s 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.

These means, business object “Item Cost” is made up of 3 entities, namely - 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendar_code</td>
<td>Y</td>
<td>A valid Calendar Code must exist in cm_cldr_mst.</td>
</tr>
<tr>
<td>period_code</td>
<td>Y</td>
<td>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, only insert is allowed. No updates and deletes are allowed.</td>
</tr>
<tr>
<td>cost_mthd_code</td>
<td>Y</td>
<td>A valid Cost Method Code must exist in cm_mthd_mst.</td>
</tr>
<tr>
<td>whse_code</td>
<td>Y</td>
<td>A valid Warehouse Code must exist in ic_whse_mst.</td>
</tr>
<tr>
<td>item_id</td>
<td>N</td>
<td>Valid Item ID (Either item_id or item_no is required). Item ID/Item No must exist in ic_item_mst.</td>
</tr>
<tr>
<td>item_no</td>
<td>N</td>
<td>Valid Item No. (Either item_id or item_no is required). Item ID/Item No must exist in ic_item_mst.</td>
</tr>
<tr>
<td>user_name</td>
<td>Y</td>
<td>Valid Oracle Application’s user name. User Name must exist in fnd_user.</td>
</tr>
</tbody>
</table>
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 very 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmpntcost_id</td>
<td>N</td>
<td>In case of “Insert”, this will be generated for each new record. Otherwise value supplied in this parameter will be used to locate unique cost detail record.</td>
</tr>
<tr>
<td>cost_cmpntcls_id</td>
<td>N</td>
<td>Valid Cost Component Class ID (Either cost_cmpntcls_id or cost_cmpntcls_code is required). Cost Component Class ID must exist in cm_cmpt_mst.</td>
</tr>
<tr>
<td>cost_cmpntcls_code</td>
<td>N</td>
<td>Valid Cost Component Class Code (Either cost_cmpntcls_id or cost_cmpntcls_code is required). Cost Component Class ID must exist in cm_cmpt_mst.</td>
</tr>
<tr>
<td>cmpnt_cost</td>
<td>Y</td>
<td>Component Cost must be a valid number.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>burden_ind</td>
<td>Y</td>
<td>Burden Indicator can be either 0 or 1. If Burden Indicator is 1 then Cost Component Class must have usage indicator = 2.</td>
</tr>
<tr>
<td>total_qty</td>
<td>N</td>
<td>Total transaction quantity used in calculating actual cost</td>
</tr>
<tr>
<td>costcalc_orig</td>
<td>N</td>
<td>This field is only a placeholder and value supplied by API caller will not be considered. For each new record created, this field will be set to 3 to indicate API origin.</td>
</tr>
<tr>
<td>Rmcalc_type</td>
<td>N</td>
<td>Raw Material Cost Calculation Type (Applicable only under actual costs, otherwise it is =0 1=PMAC, 2=PWAC, 3=PPAC, 4=LSTT, 5=LSTI</td>
</tr>
<tr>
<td>Delete_mark</td>
<td>Y</td>
<td>Delete Mark must be either 0 or 1. Insert API will ignore the value sent. New records will always be created with Delete Mark set to 0. Record cannot be deleted using Update API (marked for purge), but can be undeleted.</td>
</tr>
<tr>
<td>attribute1-30</td>
<td>N</td>
<td>Descriptive Flexfield Segment. Attribute1 through Attribute30 and Attribute Category are optional and can have alphanumeric data.</td>
</tr>
<tr>
<td>attribute_category</td>
<td>N</td>
<td>Descriptive Flexfield Segment. Attribute1 through Attribute30 and Attribute Category are optional and can have alphanumeric data.</td>
</tr>
</tbody>
</table>

**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:
Parameter - x_costcmpnt_ids (OUT)
This is a table type parameter, which has multiple records of newly generated surrogates for cost component records. API Caller will receive this OUT parameter with cost component IDs for each cost record both at this level and lower level. Following table explains how these IDs will be returned:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost_cmpntcls_id</td>
<td>Cost Component Class ID</td>
</tr>
<tr>
<td>cost_analysis_code</td>
<td>Cost Analysis Code</td>
</tr>
<tr>
<td>cost_level</td>
<td>Cost Level where the record is created</td>
</tr>
<tr>
<td>cmpntcost_id</td>
<td>Surrogate generated for combination of Cost Component Class ID+ Cost Analysis Code+Cost Level</td>
</tr>
</tbody>
</table>

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 will be available to users for use.
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 :=
}
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
    );
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 above parameters please refer to the Structure and Logic topic in this guide.

The Public API will perform 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 will insert component cost details by calling necessary routines. API will generate surrogate key - cmpntcost_id for each component cost detail begin inserted. Also, API will return a parameter with cmpntcost_ids for each newly created component cost record.
Item Cost

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 be updated. These attributes will be 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 above parameters please refer to the Structure and Logic topic in this guide.

Here second and third parameter holds surrogate - cmpntcost_id for the Item Cost record. If a valid value is supplied for this attribute then it will be used to identify the record to be updated, otherwise system will look 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. Same is true for p_lower_level_dtl_tbl records.

Note: During 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 (i.e., after the validations to all detail records) if insert fails for any reason then all the detail records for the item in process will not be inserted. Error message will be given showing the details of the record for which insert failed.

Only one message will be given for number of rows successfully inserted for an Item.
If any column must be updated to NULL, then you must pass in FND_APLG_MISS_CHAR, FND_APLG_MISS_NUM, or FND_APLG_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 be deleted. These attributes will be supplied through following two 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

Here second and third parameter holds surrogate - cmpntcost_id for the Item Cost record. If valid value is supplied for this attribute then it will be used to identify the record to be deleted, otherwise system will look 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. Same is true for p_lower_level_dtl_tbl.

Note: For details on above parameters please refer to the Structure and Logic topic in this guide.
**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 be retrieved. These attributes will be supplied through following parameter:

- **p_header_rec** Header level Information for Item Cost record

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

Item Costs will be returned back to caller using following OUT parameters:

- **x_this_level_dtl_tbl** This level cost details
- **x_lower_level_dtl_tbl** Lower level cost details

**Note:** During 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 will not be deleted. Error message will be given showing the details of the record for which delete failed.

Only one message will be given for number of rows successfully deleted for an Item.
Resource Cost

Resource Cost specifies cost of a given resource per unit of usage. This cost will be added to 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 is fairly simple and it maps to single entity which we will call 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resources</td>
<td>Y</td>
<td>Valid Resource Code must exist in cr_rsrc_mst</td>
</tr>
<tr>
<td>orgn_code</td>
<td>Y</td>
<td>Valid Organization Code must exist in sy_orgn_mst</td>
</tr>
<tr>
<td>calendar_code</td>
<td>Y</td>
<td>Valid Calendar Code must exist in cm_cldr_mst.</td>
</tr>
<tr>
<td>period_code</td>
<td>Y</td>
<td>Valid Period code must exist in cm_cldr_dtl and must not be closed.</td>
</tr>
<tr>
<td>cost_mthd_code</td>
<td>Y</td>
<td>Valid Cost Method Code must exist in cm_mthd_mst</td>
</tr>
<tr>
<td>usage_um</td>
<td>Y</td>
<td>Valid Unit of Measure must exist in sy_uoms_mst</td>
</tr>
<tr>
<td>nominal_cost</td>
<td>Y</td>
<td>Valid Number must be a number &gt; 0</td>
</tr>
<tr>
<td>delete_mark</td>
<td>N</td>
<td>Either 0 or 1. (Default is 0)</td>
</tr>
</tbody>
</table>
### 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 will be available to users for use.

```plaintext
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
); 
```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_name</td>
<td>Y</td>
<td>Valid Oracle Application’s user name. User name must exist in fnd_user</td>
</tr>
</tbody>
</table>
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 above parameters please refer to the **Structure and Logic** topic in this guide.

The Public API will perform 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 will insert 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 be updated. These attributes will be supplied through following parameter:

p_resource_cost_rec Information for Resource Cost record

---

Note: For details on above parameters please 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 Resource Cost record, API caller needs to specify attributes which uniquely classify the individual record to be deleted. These attributes will be supplied through following parameters:

p_resource_cost_rec Information for Resource Cost record

---

Note: For details on above parameters please 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, API caller needs to specify attributes which uniquely classify which Resource costs needs to be retrieved. These attributes will be supplied through following parameters:

p_resource_cost_rec Information for Resource Cost record
Resource Costs will be returned back to caller using following OUT parameters:

\[ x_{resource\_cost\_rec} \] Information for Resource Cost record

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

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

Allocation Definition Entity

Allocation Definition is fairly simple and it maps to single entity which we will call 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alloc_id</td>
<td>Y</td>
<td>Surrogate for Allocation Code (Either alloc_id OR co_code + alloc_code is required.)</td>
</tr>
<tr>
<td>co_code</td>
<td>Y</td>
<td>Valid Company Code (Either alloc_id OR co_code + alloc_code is required.)</td>
</tr>
<tr>
<td>alloc_code</td>
<td>Y</td>
<td>Valid Allocation Code (Either alloc_id OR co_code + alloc_code is required.)</td>
</tr>
<tr>
<td>alloc_method</td>
<td>Y</td>
<td>Valid Allocation Method (Either 0 or 1)</td>
</tr>
</tbody>
</table>
This section explains structure and logic of public APIs for Allocation Definition.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>line_no</td>
<td>N</td>
<td>Surrogate for detail record. Only used in case of Update operation.</td>
</tr>
<tr>
<td>item_id</td>
<td>Y</td>
<td>Surrogate for Item No. (At least one of these two parameters is required.)</td>
</tr>
<tr>
<td>item_no</td>
<td>Y</td>
<td>valid Item No. (At least one of these two parameters is required.)</td>
</tr>
<tr>
<td>Basis_account_key</td>
<td>Y</td>
<td>A valid Basis Account Key (Used only when Allocation Method = 0).</td>
</tr>
<tr>
<td>balance_type</td>
<td>Y</td>
<td>Balance Type must be 0 = Statistical; 1 = Budget; or 2 = Actual. (Used only when Allocation Method = 0).</td>
</tr>
<tr>
<td>bas_ytd_ptd</td>
<td>Y</td>
<td>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).</td>
</tr>
<tr>
<td>fixed_percent</td>
<td>Y</td>
<td>Fixed percentage must be a valid number between 1 and 100 (Used only when Allocation Method = 0).</td>
</tr>
<tr>
<td>cmpntcls_id</td>
<td>Y</td>
<td>Surrogate for Cost Component Class (At least one of these two parameters is required.)</td>
</tr>
<tr>
<td>cost_cmpntcls_code</td>
<td>Y</td>
<td>Valid Cost Component Class (At least one of these two parameters is required.)</td>
</tr>
<tr>
<td>analysis_code</td>
<td>Y</td>
<td>Valid Analysis Code must exist in cm_alys_mst</td>
</tr>
<tr>
<td>whse_code</td>
<td>Y</td>
<td>Valid Warehouse Code must exist in ic_whse_mst</td>
</tr>
<tr>
<td>delete_mark</td>
<td>N</td>
<td>Either 0 or 1. (Default is 0)</td>
</tr>
<tr>
<td>user_name</td>
<td>Y</td>
<td>Valid Oracle Application’s user name. User name must exist in fnd_user.</td>
</tr>
</tbody>
</table>

**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 will be available to users for use.

```plsql
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 );
```

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 above parameters please refer to the Structure and Logic topic in this guide.

The Public API will perform 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 will insert 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 be updated. These attributes will be supplied through following parameter:

- **p_allocation_definition_rec** Information for Allocation Definition record
Note: For details on above parameters please refer to the Structure and Logic topic in this guide.

Following are the attributes which will be used to uniquely identify which record needs to be updated:

- ALLOC_ID or ALLOC_CODE + CO_CODE
- LINE_NO

If any column must be updated to NULL, then you must pass in FND_APIG_MISS_CHAR, FND_APIG_MISS_NUM, or FND_APIG_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 Allocation Definition record, API caller needs to specify attributes which uniquely classify the individual record to be deleted. These attributes will be supplied through following parameters:

**p_allocation_definition_rec** Information for Allocation Definition record

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

Following are the attributes which will be used to uniquely identify which record needs to be deleted:

- ALLOC_ID or ALLOC_CODE + CO_CODE
- LINE_NO
Burden Details

Burden Details 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 will be 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 following attributes of business object - Burden Details:

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

Following is the definition of Header_Rec_Type:

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

Detail Entity specifies exact resource with which burden will be associated along with component, analysis code, Burden Usage, Item Quantity and Item UOM. Along with that 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>burdenline_id</td>
<td>N</td>
<td>In case of “Insert”, this will be generated for each new record. Otherwise value supplied in this parameter will be used to locate unique burden detail record.</td>
</tr>
<tr>
<td>resources</td>
<td>Y</td>
<td>Valid Resource Code must exist in cr_rsrc_mst</td>
</tr>
<tr>
<td>cost_cmpntcls_id</td>
<td>N</td>
<td>Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst and must have usage indicator = 2 (Burden/Overhead).</td>
</tr>
<tr>
<td>cost_cmpntcls_code</td>
<td>N</td>
<td>Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst and must have usage indicator = 2 (Burden/Overhead).</td>
</tr>
<tr>
<td>cost_analysis_code</td>
<td>Y</td>
<td>Valid Cost Analysis Code must exist in cm_alys_mst.</td>
</tr>
<tr>
<td>burden_usage</td>
<td>Y</td>
<td>Valid Number between 0.000000001 and 999999999.999999999</td>
</tr>
</tbody>
</table>


**Parameter - x_burdenline_ids (OUT)**

This is a table type parameter, which is having multiple records of newly generated surrogates for burden detail records. API Caller will receive this OUT parameter with BurdenLine IDs for each Burden Detail record. Following table explains how these IDs will be returned:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>Resource used for the burden cost</td>
</tr>
<tr>
<td>cost_cmpntcls_id</td>
<td>Cost Component Class ID</td>
</tr>
<tr>
<td>cost_analysis_code</td>
<td>Cost Analysis Code</td>
</tr>
<tr>
<td>burdenline_id</td>
<td>Surrogate generated for combination of Resource+Cost Component Class ID+ Cost Analysis Code</td>
</tr>
</tbody>
</table>

**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 will be available to users for use.

```
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_code cm_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;
```

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,
Public Procedure Create_Burden_Details
Procedure Create_Burden_Details is used to insert standard resource burdens. API caller needs to supply 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 above parameters please refer to the Structure and Logic topic in this guide.

The Public API will perform 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 will insert burden details by calling necessary routines. API will generate surrogate key - burdenline_id for each burden details row begin inserted. Also, API will return a parameter with burdenline_ids for each newly created burden detail record.
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 be updated. These attributes will be supplied through following two parameters along with the other details which needs to be updated:

- **p_header_rec** Header level Information for Burden Details record
- **p_dtl_tbl** Detail level Information for Burden Details record

---

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

Here second parameter holds surrogate - burdenline_ids for the Burden Cost record. If valid value is supplied for this attribute then it will be used to identify the record to be updated, otherwise system will look 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 needs to be updated.

---

**Note:** During 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 (i.e., after the validations to all detail records) if insert fails for any reason then all the detail records for the item in process will not be inserted. Error message will be given showing the details of the record for which insert failed.

Only one message will be given for number of rows successfully inserted 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_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 will be supplied through following two parameters along with the exact cost which needs to be deleted:

- **p_header_rec**  Header level Information for Burden Details record
- **p_dtl_tbl**  Detail level Information for Burden Details record

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

Here second parameter holds surrogate - burdenline_id for the burden details record. If valid value is supplied for this attribute then it will be used to identify the record to be deleted, otherwise system will look 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 needs to be deleted.

---

**Note:**  During 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 will not be updated. Error message will be given showing the details of the record for which update failed.

Only one message will be 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 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 be retrieved. These attributes will be supplied through following parameter:

p_header_rec Header level Information for Burden Details record

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

Burden Details will be returned back to caller using following OUT parameters:

x_dtl_tbl Burden details
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.
Handling Messages

**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 should 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. They should 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 should use the procedure FND_MSG_PUB.Get. This procedure operates in 5 different 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

For the sake of performance and reduction of the overall number of calls a program needs to make in order to execute an API. 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_data OUT VARCHAR2
```

**A-2**  Product Title/BookTitle as a Variable
Example:

PROCEDURE Create_OrderLine
  ( p_api_version IN NUMBER,
    p_init_msg_list INVARCHAR2 := FND_API.G_FALSE,
    p_commit IN VARCHAR2 := FND_API.G_FALSE,
    p_validation_level IN NUMBER :=
      FND_API.G_VALID_LEVEL_FULL,
    x_return_status OUT VARCHAR2,
    x_msg_count OUT NUMBER,
    x_msg_data OUT VARCHAR2,
    p_line_rec IN Line_Rec_Type
  ) IS
  l_api_version CONSTANT NUMBER := 1.0;
  l_api_name CONSTANT VARCHAR2(30) := 'Create_OrderLine';
  BEGIN
  -- Standard begin of API savepoint
  SAVEPOINT Create_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
A-4  Product Title/BookTitle as a Variable

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 will be output to the stored procedure message file, and can be 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 will be retrieved in US English.

The following is a complete listing of Cost Management API Error Messages. Note that a message that is preceded with Warning is not a fatal API error, just a warning, and 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 will be populated at runtime.
<table>
<thead>
<tr>
<th>Message Name</th>
<th>Message Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error: Burden Details delete failed for Resource &amp;RESOURCE, Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_BRDN_DEL_FAILED_DTLS</td>
</tr>
<tr>
<td>Error: Either Allocation Id or combination of Allocation Code and Company Code is required.</td>
<td>GMF_API_ALLOC_DTL_REQ</td>
</tr>
<tr>
<td>Error: Allocation Method is required.</td>
<td>GMF_API_ALLOC_MTHD_REQ</td>
</tr>
<tr>
<td>Error: Analysis Code is required.</td>
<td>GMF_API_ANALYSIS_CODE_REQ</td>
</tr>
<tr>
<td>Error: Balance Type is required.</td>
<td>GMF_API_BALANCE_TYPE_REQ</td>
</tr>
<tr>
<td>Error: Basis Account Key is required.</td>
<td>GMF_API_BAS_ACC_KEY_REQ</td>
</tr>
<tr>
<td>Error: YTD/PTD indicator for balance is required.</td>
<td>GMF_API_BAS_YTD_PTD_REQ</td>
</tr>
<tr>
<td>Error: Burden Details delete failed for Burden line Id &amp;BURDENLINE_ID.</td>
<td>GMF_API_BRDN_DEL_FAILED_ID</td>
</tr>
<tr>
<td>Burden details not found for Resource &amp;RESOURCE, Component Class Code &amp;CMPNTCLS_CODE, Component Class Id &amp;CMPNTCLS_ID, and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_BRDN_DTL_NOT_FOUND</td>
</tr>
<tr>
<td>Error: Burden Details insert failed for Resource &amp;RESOURCE, Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_BRDN_INSERT_FAILED</td>
</tr>
<tr>
<td>Error: No Burden Details found to update for Resource &amp;RESOURCE, Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_BRDN_NOT_FOUND_FOR_DTL</td>
</tr>
<tr>
<td>Error: No Burden Details found to update for Burden Line Id &amp;BURDENLINE_ID.</td>
<td>GMF_API_BRDN_NOT_FOUND_FOR_ID</td>
</tr>
</tbody>
</table>
### Understanding Error Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Message Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>UOM Conversion failed for resource &amp;RESOURCES from Burden UOM &amp;BURDEN_UM to Resource UOM &amp;BURDEN_UM.</td>
<td>GMF_API_BRDN_UM_CONV_ERR</td>
</tr>
<tr>
<td>Error: Burden Details update failed for Resource &amp;RESOURCE, Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_BRDN_UPD_FAILED_DTLS</td>
</tr>
<tr>
<td>Error: Burden Details update failed for Burden line Id &amp;BURDENLINE_ID.</td>
<td>GMF_API_BRDN_UPD_FAILED_ID</td>
</tr>
<tr>
<td>Error: Burden Indicator is required.</td>
<td>GMF_API_BURDEN_IND_REQ</td>
</tr>
<tr>
<td>Error: Burden Quantity is required.</td>
<td>GMF_API_BURDEN_QTY_REQ</td>
</tr>
<tr>
<td>Error: Burden UOM is required.</td>
<td>GMF_API_BURDEN_UM_REQ</td>
</tr>
<tr>
<td>Error: Burden Indicator supplied is &amp;BURDEN_IND, but Usage of the Cost Component Class &amp;CMPNT_CLS is not burden detail.</td>
<td>GMF_API_BURDEN_USAGE_IND</td>
</tr>
<tr>
<td>Error: Burden Usage is required.</td>
<td>GMF_API_BURDEN_USAGE_REQ</td>
</tr>
<tr>
<td>Error: Calendar Code is required.</td>
<td>GMF_APICALENDAR_CODE_REQ</td>
</tr>
<tr>
<td>Error: Record cannot be marked for purge using Update API.</td>
<td>GMF_API_CANT_MARK_FOR_PURGE</td>
</tr>
<tr>
<td>Error: Period &amp;PERIOD_CODE for Calendar &amp;CALENDAR_CODE is Closed.</td>
<td>GMF_API_CLOSED PERIOD</td>
</tr>
<tr>
<td>Error: Component Cost is required.</td>
<td>GMF_API_CMPNT_COST_REQ</td>
</tr>
<tr>
<td>Error: Either Component Class Id or Component Class Code is required.</td>
<td>GMF_API_CMPNTCLS_ID_REQ</td>
</tr>
<tr>
<td>Error: Usage of Component Class &amp;CMPNTCLS is not GL Expense Allocation.</td>
<td>GMF_API_CMPNTCLS_USG_NOT_ALC</td>
</tr>
<tr>
<td>Error: Usage of Component Class &amp;CMPNTCLS_ID is not Burden (Overhead).</td>
<td>GMF_API_CMPNTCLS_USG_NOT_BRDN</td>
</tr>
<tr>
<td>Error: Cost Method Code is required.</td>
<td>GMF_API_COST_MTHD_CODE_REQ</td>
</tr>
<tr>
<td>Message Name</td>
<td>Message Code</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>Error: Item Cost details not found for Item Id &amp;ITEM_ID, Warehouse &amp;WHSE_CODE, Calendar &amp;CALENDAR_CODE, Period &amp;PERIOD_CODE, Cost Method &amp;COST_MTHD, Component Class &amp;CMPNTCLS_ID, Analysis Code &amp;ANALYSIS_CODE and Cost Level &amp;COST_LEVEL.</td>
<td>GMF_API_COST_ROWS_NOT_FOUND</td>
</tr>
<tr>
<td>DEBUG: &amp;MSG</td>
<td>GMF_API_DEBUG</td>
</tr>
<tr>
<td>Error: No Burden Details found to delete for Resource &amp;RESOURCE, Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_DEL_BRDN_NOT_FOUND_DTL</td>
</tr>
<tr>
<td>Error: No Burden Details found to delete for Burden Line Id &amp;BURDENLINE_ID.</td>
<td>GMF_API_DEL_BRDN_NOT_FOUND_ID</td>
</tr>
<tr>
<td>Error: No Component Cost Details found to delete for Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_DEL_IC_NOT_FOUND_DTL</td>
</tr>
<tr>
<td>Error: No Component Cost Details found to delete for Component Cost Id &amp;CMPNTCOST_ID.</td>
<td>GMF_API_DEL_IC_NOT_FOUND_ID</td>
</tr>
<tr>
<td>Error: Delete Mark is required.</td>
<td>GMF_API_DELETE_MARK_REQ</td>
</tr>
<tr>
<td>Error: Percentage is required.</td>
<td>GMF_API_FIXED_PERCENT_REQ</td>
</tr>
<tr>
<td>Error: No Updates/Deletes allowed since Period &amp;PERIOD_CODE for Calendar &amp;CALENDAR_CODE is Frozen.</td>
<td>GMF_API_FROZEN_PERIOD</td>
</tr>
</tbody>
</table>
## Understanding Error Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Message Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error: Cannot insert new component details since Item &amp;ITEM is frozen for Whse &amp;WHSE, Calendar &amp;CALENDAR, Period &amp;PERIOD and Cost Method &amp;COST_MTHD.</td>
<td>GMF_API_IC_CANNT_INSERT_CMPTS</td>
</tr>
<tr>
<td>Error: Component Cost delete failed for Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_IC_DEL_FAILED_DTLS</td>
</tr>
<tr>
<td>Error: Component Cost delete failed for Component Cost Id &amp;CMPNTCOST_ID.</td>
<td>GMF_API_IC_DEL_FAILED_ID</td>
</tr>
<tr>
<td>Error: Lower Level Component Cost insert failed for Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_IC_LWRLVL_INS_FAILED</td>
</tr>
<tr>
<td>Error: This Level Component Cost insert failed for Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_IC_THISLVL_INS_FAILED</td>
</tr>
<tr>
<td>Error: Component Cost update failed for Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_IC_UPD_FAILED_DTLS</td>
</tr>
<tr>
<td>Error: Component Cost update failed for Component Cost Id &amp;CMPNTCOST_ID.</td>
<td>GMF_API_IC_UPD_FAILED_ID</td>
</tr>
<tr>
<td>Warning: Allocation Code &amp;ALLOC_CODE and Company Code &amp;CO_CODE values are ignored since allocation Id is supplied.</td>
<td>GMF_API_IGNORE_ALLOC_CODE</td>
</tr>
<tr>
<td>Warning: Basis Amount fields are ignored since fixed percent is used for Allocation (Allocation Method is Fixed Percent).</td>
<td>GMF_API_IGNORE_BASIS</td>
</tr>
<tr>
<td>Warning: BurdenLine Id &amp;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.</td>
<td>GMF_API_IGNORE_BRDN_UNIQUE_KEY</td>
</tr>
<tr>
<td>Warning: Component Class Code &amp;CMPNTCLS_CODE is ignored since Component Class Id is supplied.</td>
<td>GMF_API_IGNORE_CMPNTCLS_CODE</td>
</tr>
<tr>
<td>Warning: Fixed Percent &amp;FIXED_PERCENT value is ignored since Alloc Percent Depends on Basis Amount (Allocation Method is Basis Type).</td>
<td>GMF_API_IGNORE_FIXED_PERCENT</td>
</tr>
<tr>
<td>Message Name</td>
<td>Message Code</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Warning: Component Cost Id &amp;CMPNTCOST_ID is supplied and will be used to</td>
<td>GMF_API_IGNORE_IC_UNIQUE_KEY</td>
</tr>
<tr>
<td>Update/Delete the record. Unique key combination of Item, Warehouse, Calendar,</td>
<td></td>
</tr>
<tr>
<td>Period, Cost Method, Component Class, Analysis Code and Cost Level will be</td>
<td></td>
</tr>
<tr>
<td>ignored.</td>
<td></td>
</tr>
<tr>
<td>Warning: Item No &amp;ITEM_NO is ignored since Item Id is supplied.</td>
<td>GMF_API_IGNORE_ITEM_NO</td>
</tr>
<tr>
<td>Warning: Lower level details cannot be updated or deleted hence ignored.</td>
<td>GMF_API_IGNORE_LOWER_LEVEL</td>
</tr>
<tr>
<td>Error: Invalid Accounting No in the Basis Account Key &amp;BAS_ACC_KEY.</td>
<td>GMF_API_INVALID_ACCT_NO</td>
</tr>
<tr>
<td>Error: Invalid Accounting Unit in the Basis Account Key &amp;BAS_ACC_KEY.</td>
<td>GMF_API_INVALID_ACCTG_UNIT</td>
</tr>
<tr>
<td>Error: Invalid combination of Allocation Code &amp;ALLOC_CODE and Company &amp;CO_CODE.</td>
<td>GMF_API_INVALID_ALLOC_CODE</td>
</tr>
<tr>
<td>Error: Allocation Method &amp;ALLOC_METHOD not consistent with the existing method.</td>
<td>GMF_API_INVALID_ALLOC_DEF</td>
</tr>
<tr>
<td>Error: Invalid Allocation Id: &amp;ALLOC_ID.</td>
<td>GMF_API_INVALID_ALLOC_ID</td>
</tr>
<tr>
<td>Error: Invalid Allocation Method: &amp;ALLOC_METHOD.</td>
<td>GMF_API_INVALID_ALLOC_MTHD</td>
</tr>
<tr>
<td>Error: Invalid Analysis Code: &amp;ANALYSIS_CODE.</td>
<td>GMF_API_INVALID_ANALYSIS_CODE</td>
</tr>
<tr>
<td>Error: Invalid Balance Type: &amp;BALANCE_TYPE.</td>
<td>GMF_API_INVALID_BALANCE_TYPE</td>
</tr>
<tr>
<td>Error: Invalid Basis amount Indicator: &amp;BAS_YTD_PTD.</td>
<td>GMF_API_INVALID_BAS_YTD_PTD</td>
</tr>
<tr>
<td>Invalid burden line id &amp;BURDENLINE_ID passed.</td>
<td>GMF_API_INVALID_BRDN_LINE_ID</td>
</tr>
<tr>
<td>Error: Invalid Burden Indicator: &amp;BURDEN_IND.</td>
<td>GMF_API_INVALID_BURDEN_IND</td>
</tr>
<tr>
<td>Error: Invalid Burden Quantity: &amp;BURDEN_QTY (Should be between 0.000000001 and 999999999D999999999).</td>
<td>GMF_API_INVALID_BURDEN_QTY</td>
</tr>
<tr>
<td>Error: Invalid Burden UOM: &amp;BURDEN_UM.</td>
<td>GMF_API_INVALID_BURDEN_UM</td>
</tr>
</tbody>
</table>
### Understanding Error Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Message Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error: Invalid Burden Usage: &amp;BURDEN_USAGE (Should be between 0.000000001 and 999999999D999999999).</td>
<td>GMF_API_INVALID_BURDEN_USAGE</td>
</tr>
<tr>
<td>Error: Invalid Calendar Code: &amp;CALENDAR_CODE.</td>
<td>GMF_API_INVALIDCALENDAR_CODE</td>
</tr>
<tr>
<td>Error: Component cost &amp;CMPNT_COST should not be greater than 999,999,999,999999999.</td>
<td>GMF_API_INVALID_CMPNT_COST</td>
</tr>
<tr>
<td>Error: Invalid Component Class Code: &amp;CMPNTCLS_CODE.</td>
<td>GMF_API_INVALID_CMPNTCLS_CODE</td>
</tr>
<tr>
<td>Error: Invalid Component Class Id: &amp;CMPNTCLS_ID.</td>
<td>GMF_API_INVALID_CMPNTCLS_ID</td>
</tr>
<tr>
<td>Error: Invalid Cost Method Code: &amp;COST_MTHD_CODE.</td>
<td>GMF_API_INVALID_COST_MTHD_CODE</td>
</tr>
<tr>
<td>Error: Invalid Delete Mark: &amp;DELETE_MARK.</td>
<td>GMF_API_INVALID_DELETE_MARK</td>
</tr>
<tr>
<td>Error: Invalid Percentage: &amp;FIXED_PERCENT (Should be 0 to 100).</td>
<td>GMF_API_INVALID_FIXED_PERCENT</td>
</tr>
<tr>
<td>Error: Invalid Item Id: &amp;ITEM_ID.</td>
<td>GMF_API_INVALID_ITEM_ID</td>
</tr>
<tr>
<td>Error: Invalid Item No: &amp;ITEM_NO.</td>
<td>GMF_API_INVALID_ITEM_NO</td>
</tr>
<tr>
<td>Error: Invalid Item Quantity: &amp;ITEM_QTY (Should be between 0.000000001 and 999999999D999999999).</td>
<td>GMF_API_INVALID_ITEM_QTY</td>
</tr>
<tr>
<td>Error: Invalid Item UOM: &amp;ITEM_UM.</td>
<td>GMF_API_INVALID_ITEM_UM</td>
</tr>
<tr>
<td>Error: Invalid Nominal Cost: &amp;NOMINAL_COST (Should be between 0 and 999999999D999999999).</td>
<td>GMF_API_INVALID_NOMINAL_COST</td>
</tr>
<tr>
<td>Error: Invalid Organization Code: &amp;ORGN_CODE.</td>
<td>GMF_API_INVALID_ORGN_CODE</td>
</tr>
<tr>
<td>Error: Invalid Period Code: &amp;PERIOD_CODE for Calendar &amp;CALENDAR_CODE.</td>
<td>GMF_API_INVALID_PERIOD_CODE</td>
</tr>
<tr>
<td>Error: Invalid Resource Code: &amp;RESOURCES.</td>
<td>GMF_API_INVALIDRESOURCES</td>
</tr>
<tr>
<td>Error: For Actual Cost method, Raw Material Cost Calculation Type should be 1-PMAC, 2-PWAC, 3-PPAC, 4-LSTT or 5 -LSTI.</td>
<td>GMF_API_INVALID_RMCALC_TYPE</td>
</tr>
<tr>
<td>Error: Invalid Usage Unit of Measure: &amp;USAGE_UM.</td>
<td>GMF_API_INVALID_USAGE_UM</td>
</tr>
</tbody>
</table>
## Understanding Error Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Message Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error: Invalid User name: &amp;USER_NAME.</td>
<td>GMF_API_INVALID_USER_NAME</td>
</tr>
<tr>
<td>Error: Invalid Warehouse Code: &amp;WHSE_CODE.</td>
<td>GMF_API_INVALID_WHSE_CODE</td>
</tr>
<tr>
<td>Error: Either Item Id or Item No is required.</td>
<td>GMF_API_ITEM_ID_REQ</td>
</tr>
<tr>
<td>Error: Item Quantity is required.</td>
<td>GMF_API_ITEM_QTY_REQ</td>
</tr>
<tr>
<td>Error: UOM conversion error for Item &amp;ITEM_ID. Cannot convert quantity from current Item UOM &amp;ITEM_UM to actual Item UOM &amp;ITEM_ACT_UM.</td>
<td>GMF_API_ITEM_UM_CONV_ERR</td>
</tr>
<tr>
<td>Error: Item UOM is required.</td>
<td>GMF_API_ITEM_UM_REQ</td>
</tr>
<tr>
<td>Error: Line Number is required.</td>
<td>GMF_API_LINE_NO_REQ</td>
</tr>
<tr>
<td>Message: No rows deleted.</td>
<td>GMF_API_NO_ROWS_DEL</td>
</tr>
<tr>
<td>Error: No rows found to update/delete for Allocation Id &amp;ALLOC_ID and Line No &amp;LINE_NO.</td>
<td>GMF_API_NO_ROWS_FOUND</td>
</tr>
<tr>
<td>Message: No rows inserted.</td>
<td>GMF_API_NO_ROWS_INS</td>
</tr>
<tr>
<td>Message: No rows updated.</td>
<td>GMF_API_NO_ROWS_UPD</td>
</tr>
<tr>
<td>Error: Nominal Cost is required.</td>
<td>GMF_API_NOMINAL_COST_REQ</td>
</tr>
<tr>
<td>Error: Organization Code is required.</td>
<td>GMF_API_ORGN_CODE_REQ</td>
</tr>
<tr>
<td>Error: Period Code is required.</td>
<td>GMF_API_PERIOD_CODE_REQ</td>
</tr>
<tr>
<td>Error: Resource Code is required.</td>
<td>GMF_API_RESOURCES_REQ</td>
</tr>
<tr>
<td>&amp;NUM_ROWS row(s) deleted.</td>
<td>GMF_API_ROWS_DEL</td>
</tr>
<tr>
<td>&amp;NUM_ROWS row(s) inserted.</td>
<td>GMF_API_ROWS_INS</td>
</tr>
<tr>
<td>&amp;NUM_ROWS row(s) updated.</td>
<td>GMF_API_ROWS_UPD</td>
</tr>
</tbody>
</table>
### Understanding Error Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Message Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error: No rows found to update/delete for Resource &amp;RESOURCES, Orgn &amp;ORGN_CODE, Calendar &amp;CALENDAR_CODE, Period &amp;PERIOD_CODE and Cost Method &amp;COST_MTHD_CODE.</td>
<td>GMF_API_RSRC_NO_ROWS_FOUND</td>
</tr>
<tr>
<td>Warning: Total Fixed Percentage for Allocation Id &amp;ALLOC_ID is not equal to 100.</td>
<td>GMF_API_TOTAL_PCT_NOTHUNDRED</td>
</tr>
<tr>
<td>Error: Burden UOM &amp;BURDEN_UM must be of the same type as the resource UOM &amp;RESOURCE_UM.</td>
<td>GMF_API_UOM_SAMETYPE_REQ</td>
</tr>
<tr>
<td>Error: No Component Cost Details found to update for Component Class Id &amp;CMPNTCLS_ID and Analysis Code &amp;ALYS_CODE.</td>
<td>GMF_API_UPD_IC_NOT_FOUND_DTL</td>
</tr>
<tr>
<td>Error: No Component Cost Details found to update for Component Cost Id &amp;CMPNTCOST_ID.</td>
<td>GMF_API_UPD_IC_NOT_FOUND_ID</td>
</tr>
<tr>
<td>Error: Usage Unit of Measure is required.</td>
<td>GMF_API_USAGE_UM_REQ</td>
</tr>
<tr>
<td>Error: Usage UOM &amp;USAGE_UM must be of the same type as the resource UOM &amp;RESOURCE_UM.</td>
<td>GMF_API_USAGE_UOM_SAMETYPE_REQ</td>
</tr>
<tr>
<td>User Name is required.</td>
<td>GMF_API_USER_NAME_REQ</td>
</tr>
<tr>
<td>Error: Warehouse Code is required.</td>
<td>GMF_API_WHSE_CODE_REQ</td>
</tr>
</tbody>
</table>
Application Programming Interface (API)
A documented, supported method for communicating within or between 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 API’s. An example of a Module is Oracle Inventory.

Business Object
An independent item of significance in the business world. An example of a “Business Object” is a Sales Order.

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”.

Business Process API
An API that performs a transaction for the calling module, e.g., to hire an employee, enter an order, or cost a material movement transaction.

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.
Group API

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

A
Allocation Definition, 1-3
    Allocation Definition Entity, 4-17
Delete, 1-5
Insert, 1-5
Structure, 3-4
  Structure and Logic, 4-18
Update, 1-5
Allocation Definition APIs, 1-5
API, 1-1, 1-2
Application Programming Interface (API), 1

B
Burden Details, 1-3
  Detail Entity, 4-23
  Header Entity, 4-22
  Insert, 1-5
  Retrieve, 1-6
  Structure, 3-5
  Structure and Logic, 4-24
  Update, 1-5
Burden Details APIs, 1-5
Business Object, 1
Business Objects, 4-1
  Allocation Definition, 4-17
  Burden Details, 4-22
  Item Cost, 4-1
  Resource Cost, 4-12
Business Process API, 1

C
Code Re-Use, 1-3
Consistent Behavior, 1-3

E
Ease of Integration, 1-3
Entity, 1

F
FND_API, 1-7
FND_APIG_FALSE, 3-6
FND_MESSAGE, 1-7
FND_PUB_MSG, 1-7
formula information, importing, 1-2

G
GMF_AllocationDefinition_PUB, 1-7, 3-4
GMF_BurdenDetails_PUB, 1-7, 3-5
GMF_ItemCost_PUB, 1-7, 3-1
GMF_ResourceCost_PUB, 1-7, 3-3
GMFALCB.pls, 3-4
GMFPALCS.pls, 1-7, 3-4
GMFPBRDB.pls, 1-7, 3-5
GMFPBRDS.pls, 1-7, 3-5
GMFPCTSTRB.pls, 1-7, 3-1
GMFPCTST.pls, 1-7, 3-1
GMFPRESSB.pls, 3-3
GMFPRESS.pls, 1-7, 3-3
Group API, 2
importing formula information, 1-2
Insulation from Changes, 1-3
Item Cost, 1-3
  Cost Header Entity, 4-2
  Delete, 1-4
  Insert, 1-4
  Lower Level Cost Detail Entity, 4-4
  Retrieve, 1-4
  Structure, 3-1
  Structure and Logic, 4-5
  This Level Cost Detail Entity, 4-3
  Update, 1-4
Item Cost APIs, 1-4

message encoding, 3-7
message list initialization, 3-6
message list, specifying number of messages added, 3-7
Module, 1

p_allocation_definition_rec, 4-20
p_api_version, 3-6
p_commit, 3-7
p_dtl_tbl, 4-28, 4-29
p_header_rec, 4-9, 4-10, 4-11, 4-28, 4-29, 4-30
p_init_msg_list, 3-6
p_lower_level_dtl_tbl, 4-9, 4-10
p_resource_cost_rec, 4-15, 4-21
p_this_level_dtl_tbl, 4-9, 4-10
Package Specification - GMFPALCS.pls, 4-19
Package Specification - GMFPBRDS.pls, 4-25
Package Specification - GMFPCSTS.pls, 4-5
Package Specification - GMFPPRESS.pls, 4-13
PL/SQL, 1-2
Private API, 1
processing standard message functions, 3-7
Public API, 1
Public Procedure Delete_Allocation_Definition, 4-21
Public Procedure Delete_Item_Cost, 4-10, 4-29
Public Procedure Delete_Resource_Cost, 4-15
Public Procedure Get_Burden_Details, 4-30
Public Procedure Get_Item_Cost, 4-11
Public Procedure Get_Resource_Cost, 4-15
Public Procedure Update_Allocation_Definition, 4-20
Public Procedure Update_Burden_Details, 4-28
Public Procedure Update_Item_Cost, 4-9
Public Procedure Update_Resource_Cost, 4-15

Resource Cost, 1-3
  Delete, 1-5
  Insert, 1-4
  Resource Cost Entity, 4-12
  Retrieve, 1-5
  Structure, 3-3
  Structure and Logic, 4-13
  Update, 1-4
Resource Cost APIs, 1-4
Robust Validation, 1-3

standard message function processing, 3-7
support policy, 1-6

Technical Overview, 3-1
  Allocation Definition, 3-4
  Burden Details, 3-5
  Item Cost, 3-1
  Resource Cost, 3-3

version compatibility, validation, 3-6

x_dtl_tbl, 4-30
x_lower_level_dtl_tbl, 4-11
x_msg_count, 3-7
x_msg_data, 3-7
x_resource_cost_rec, 4-16
x_return_status, 3-7
x_this_level_dtl_tbl, 4-11