

Oracle® Process Manufacturing

Cost Management API User's Guide

Release 11i

September 2001

Part No. A95270-01

ORACLE®

Part No. A95270-01

Copyright © 1996, 2001, Oracle Corporation. All rights reserved.

Primary Author: Michele-Andrea Fields

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

Program Documentation is licensed for use solely to support the deployment of the Programs and not for any other purpose.

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

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark, and Oracle Process Manufacturing is a trademark or registered trademark of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	vii
Preface.....	ix
Intended Audience	ix
About This Guide	x
Information Sources	x
Conventions.....	xiii
Do Not Use Database Tools to Modify Oracle Applications Data	xiv
Documentation Accessibility	xiv
Accessibility of Code Examples in Documentation.....	xv
About Oracle	xv
Thank You.....	xv
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
Allocation Definition APIs	1-5
Burden Details APIs	1-5
Cost Management API Features.....	1-6
Cost Management API Support Policy.....	1-6
Oracle Applications Packages Supplied	1-7

Cost Management API Bill of Materials	1-7
2 Cost Management API Usage	
Calling the API Interface Code	2-1
Calling the API Code - Example	2-2
Wrapper logic.....	2-2
Item Cost Example.....	2-3
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.....	4-16
Allocation Definition Entity	4-16
Structure and Logic	4-17
Burden Details.....	4-21
Header Entity	4-21
Detail Entity.....	4-22

Structure and Logic	4-23
---------------------------	------

A Messages and Errors

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

Index

Send Us Your Comments

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

Part No. A95270-01

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

Welcome to the *Oracle Process Manufacturing Cost Management API User's Guide*. This user's guide includes the information you need to work with the Oracle Process Manufacturing (OPM) application effectively.

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

Intended Audience

This guide assumes that you have working knowledge of your business area's processes and tools. It also assumes that you are familiar with OPM Cost Management. If you have never used OPM, we suggest you attend one or more of the Oracle Process Manufacturing training classes available through Oracle World Wide Education.

This guide also assumes that you are familiar with the Oracle Applications graphical user interface. To learn more about Oracle Applications graphical user interface, read the *Oracle Applications User's Guide*.

About This Guide

This guide contains overviews as well as task and reference information. It includes the following:

Name	Description
Cost Management API Introduction	Describes the Application Program Interfaces (APIs) that support external interfaces to the OPM Cost Management tables.
Cost Management API Usage	Provides information on the usage and layout of the Cost Management APIs.
Cost Management API Business Objects	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.
Glossary	Provides definitions of terms that are used in this guide.

Information Sources

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

Online Documentation

Oracle Applications documentation is available on CD-ROM, except for technical reference manuals. User's guides are available in HTML format and on paper. Technical reference manuals are available on paper only. Other documentation is available on paper and sometimes in PDF format.

The content of the documentation remains the same from format to format. Slight formatting differences could occur due to publication standards, but such differences do not affect content. For example, page numbers are included on paper, but are not included in HTML.

The HTML documentation is available from all Oracle Applications windows. Each window is programmed to start your web browser and open a specific,

context-sensitive section. Once any section of the HTML documentation is open, you can navigate freely throughout all Oracle Applications documentation.

Related Documents

Oracle Process Manufacturing shares business and setup information with other Oracle products. You may find the following Oracle Applications user's guides useful:

- *Oracle Applications User's Guide*
- *Oracle Application's Flexfields Guide*
- *Oracle Workflow User Guide*
- *Oracle Applications System Administrator's Guide*
- *Oracle General Ledger User's Guide*
- *Oracle Payables User's Guide*
- *Oracle Receivables User's Guide*
- *Oracle Human Resources North American User's Guide*
- *Oracle Purchasing User's Guide*

Oracle Process Manufacturing Guides

The following is a list of documentation in each product group for OPM:

Financials

- *Oracle Process Manufacturing Accounting Setup User's Guide*
- *Oracle Process Manufacturing Cost Management User's Guide*
- *Oracle Process Manufacturing Manufacturing Accounting Controller User's Guide*
- *Oracle Process Manufacturing and Oracle Financials Integration User's Guide*

Inventory Control

- *Oracle Process Manufacturing Inventory Management User's Guide*
- *Oracle Process Manufacturing Physical Inventory User's Guide*

Logistics

- *Oracle Process Manufacturing Order Fulfillment User's Guide*
- *Oracle Process Manufacturing Purchase Management User's Guide*

Process Execution

- *Oracle Process Manufacturing Process Operation Control User's Guide*
- *Oracle Process Manufacturing Production Management User's Guide*

Process Planning

- *Oracle Process Manufacturing Integration with Advanced Planning and Scheduling User's Guide*
- *Oracle Process Manufacturing MPS/MRP and Forecasting User's Guide*

Product Development

- *Oracle Process Manufacturing New Product Development User's Guide*
- *Oracle Process Manufacturing Quality Management User's Guide*

Regulatory

- *Oracle Process Manufacturing Regulatory Management User's Guide*

System Administration and Technical Reference

- *Oracle Process Manufacturing Implementation Guide*
- *Oracle Process Manufacturing System Administration User's Guide*
- Oracle Process Manufacturing Technical Reference Manuals
- Oracle Process Manufacturing API User's Guides

Training

Oracle offers a complete set of formal training courses to help you master Oracle Process Manufacturing and reach full productivity quickly. We organize these courses into functional learning paths, so you take only those courses appropriate to your area of responsibility.

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

Conventions

The following conventions are used in this guide:

Bolded Text

Buttons, fields, keys, menus, and selections are bolded in procedures only. For example: To access the next window, click **OK**. Otherwise, references to these features appear in regular type.

Additional Menu Options

Only nonstandard menu options are discussed. Standard menu bar options (such as Save) are not discussed. These standard options are described in the *Oracle Applications User's Guide*. Only menu options unique to the use of the specific window are discussed.

Field References

References to fields within procedures are in bold type. References within the body of this guide appear in regular type.

Required Fields

The word Required appears as the last word in the field description of all required fields. When the field is required contingent on the entry in another field, or only in specific situations, "Required if..." is the last sentence of the field description.

Fields Reserved for Future Use

Fields with no current processing implications are referenced by the statement "This field is not currently used" or "Reserved for future use." Do not use these fields for your own reference data, because there are plans to link future functionality to these fields. Fields intended for informational purposes only are referenced by the statement "This field is for informational purposes only."

Pending/Completed Transactions

Discussions about processing transactions that use the words pending and completed refer to the status of a transaction. Pending and completed do not refer to the database tables that are updated as a result of transactions (for example, some completed transactions are stored in the Pending Transactions table).

Procedures

Most topics contain a procedure with numbered steps. Any actions which are subordinate to a step are assigned letters. You can customize your Oracle Application, therefore, all procedures are suggestive only. Navigate to windows

and between responsibilities in a way that works best for your particular setup. Also note that fields may appear in a different order than they are discussed.

Use of the Word Character

The word character means an alphanumeric character. Characters that are numeric or alphabetic only are referenced specifically. Depending on your system security profile, you may not have access to all of the windows and functions described in this guide. If you do not see a menu option described in this guide, and you want access to it, contact your System Administrator.

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle Applications tables are interrelated. As a result, any change you make using Oracle Applications can update many tables at once. If you modify the Oracle Applications data using anything other than Oracle Applications, you could change a row in one table without making corresponding changes in related tables. If your tables are not synchronized with each other, you risk retrieving erroneous information and receiving 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 track who changes information. If you enter information into database tables using database tools, you could store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

Consequently, we strongly recommend that you never use SQL*Plus or any other tool to modify Oracle Applications data unless otherwise instructed by Oracle Support Services.

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.

About Oracle

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

Oracle Products are available for mainframes, minicomputers, personal computers, network computers, and personal digital assistants, allowing organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified 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 140 countries around the world.

Thank You

Thank you for choosing Oracle Process Manufacturing and this user's guide.

We value your comments and feedback. At the beginning of this guide is a Reader's Comment Form that you can use to explain what you like or dislike about 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 an electronic mail message to appsdoc@us.oracle.com

Cost Management APIs

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

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

Introducing the Cost Management APIs

Cost Management APIs 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.

- 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:**

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.

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

Cost Management API Usage

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


```

14  --|   It reads item data from a flat file and outputs any error
|
15  --|   messages to a second flat file. It also generates a log file
|
16  --|   called gmf_api_cric_wrapper<session_id>.log in the p_dir directory.
|
17  --|
|
18  --| PARAMETERS
|
19  --|   p_dir           IN VARCHAR2       - Working directory for input
|
20  --|                                     and output files.
|
21  --|   p_input_file    IN VARCHAR2       - Name of input file
|
22  --|   p_output_file   IN VARCHAR2       - Name of output file
|
23  --|   p_delimiter     IN VARCHAR2       - Delimiter character
|
24  --|
|
25  --| RETURNS
|
26  --|   VARCHAR2 - 'S' All records processed successfully
|
27  --|               'E' 1 or more records errored
|
28  --|               'U' 1 or more record unexpected error
|
29  --|
|
30  --| HISTORY
|
31  --|
|
32
-- +-----+
33  --Api end of comments
34
35  FUNCTION Create_Item_Cost
36  ( p_dir           IN VARCHAR2
37  , p_input_file    IN VARCHAR2
38  , p_output_file   IN VARCHAR2
39  , p_delimiter     IN VARCHAR2

```

```
40 )
41 RETURN VARCHAR2
42 IS
43
44 /* Local variables */
45
46 l_status          VARCHAR2(11);
47 l_return_status   VARCHAR2(11) :=FND_API.G_RET_STS_SUCCESS;
48 l_count           NUMBER(10) ;
49 l_record_count    NUMBER(10) :=0;
50 l_loop_cnt        NUMBER(10) :=0;
51 l_dummy_cnt       NUMBER(10) :=0;
52 l_data            VARCHAR2(1000);
53
54 l_p_dir           VARCHAR2(150);
55 l_output_file     VARCHAR2(120);
56 l_outfile_handle  UTL_FILE.FILE_TYPE;
57 l_input_file      VARCHAR2(120);
58 l_infile_handle   UTL_FILE.FILE_TYPE;
59 l_line            VARCHAR2(1000);
60 l_delimiter       VARCHAR(11);
61 l_log_dir         VARCHAR2(150);
62 l_log_name        VARCHAR2(120) :='gmf_api_cric_wrapper';
63 l_log_handle      UTL_FILE.FILE_TYPE;
64 l_global_file     VARCHAR2(120);
65
66 l_header_rec      GMF_ItemCost_PUB.Header_Rec_Type;
67 l_this_lvl_tbl    GMF_ItemCost_PUB.This_Level_Dtl_Tbl_Type;
68 l_lower_lvl_tbl   GMF_ItemCost_PUB.Lower_Level_Dtl_Tbl_Type;
69 l_costcmpnt_ids   GMF_ItemCost_PUB.costcmpnt_ids_tbl_type;
70 l_idx             NUMBER(10);
71 l_idx1            NUMBER(10);
72 l_type            NUMBER(10);
73 l_continue        VARCHAR2(1) := 'Y' ;
74 l_skip_details    VARCHAR2(1) := 'N' ;
75
76 l_session_id      VARCHAR2(110);
77
78 BEGIN
79
80 /* Enable The Buffer */
81 DBMS_OUTPUT.ENABLE(1000000);
82
83 l_p_dir           :=p_dir;
84 l_input_file      :=p_input_file;
```

```

85  l_output_file      :=p_output_file;
86  l_delimiter       :=p_delimiter;
87  l_global_file     :=l_input_file;
88
89  /* Obtain The SessionId To Append To wrapper File Name. */
90
91  l_session_id := USERENV('sessionid');
92
93  l_log_name := CONCAT(l_log_name,l_session_id);
94  l_log_name := CONCAT(l_log_name,'.log');
95
96  /* Directory is now the same same as for the out file */
97  l_log_dir := p_dir;
98
99
100 /* Open The Wrapper File For Output And The Input File for Input. */
101
102 l_log_handle      :=UTL_FILE.FOPEN(l_log_dir, l_log_name, 'w');
103 l_infile_handle  :=UTL_FILE.FOPEN(l_p_dir, l_input_file, 'r');
104
105 /* Loop thru flat file and call Item Cost API */
106 DBMS_OUTPUT.PUT_LINE('Process Started at ' || to_char(SYSDATE,'DD-MON-YY
HH24:MI:SS'));
107 DBMS_OUTPUT.PUT_LINE('Input Directory ' || l_p_dir );
108 DBMS_OUTPUT.PUT_LINE('Input File ' || l_input_file );
109 DBMS_OUTPUT.PUT_LINE('Delimiter ' || l_delimiter );
110 DBMS_OUTPUT.PUT_LINE('Output File ' || l_output_file );
111
112
113 DBMS_OUTPUT.PUT_LINE('Start Processing');
114 UTL_FILE.PUT_LINE(l_log_handle, 'Process Started at ' || to_
char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
115
116 UTL_FILE.NEW_LINE(l_log_handle);
117 UTL_FILE.PUT_LINE(l_log_handle, 'Input Directory ' || l_p_dir );
118 UTL_FILE.PUT_LINE(l_log_handle, 'Input File ' || l_input_file );
119 UTL_FILE.PUT_LINE(l_log_handle, 'Record Type ' || l_delimiter );
120 UTL_FILE.PUT_LINE(l_log_handle, 'Output File ' || l_output_file );
121
122 l_outfile_handle :=UTL_FILE.FOPEN(l_p_dir, l_output_file, 'w');
123
124 /* Get the first record from the file */
125 BEGIN
126     UTL_FILE.GET_LINE(l_infile_handle, l_line);
127     l_record_count :=l_record_count+1;

```

```

128     l_type     := Get_Field(l_line,l_delimiter,1) ; -- 10 : header rec, 20 :
this level, 30 : lower level
129     EXCEPTION
130         WHEN NO_DATA_FOUND THEN
131             raise;
132     END;
133
134     /* Populate appropriate pl/sql record or table based on the type of record
*/
135     LOOP
136         BEGIN
137
138             UTL_FILE.PUT_LINE(l_log_handle, 'Reading Record...' || l_record_count
);
139             UTL_FILE.PUT_LINE(l_log_handle, 'Type     = ' || l_type) ;
140
141             /* Header record */
142
143             IF l_type = 10 THEN
144                 -- empty the pl/sql tables to remove details of the previous item
145                 -- and initialize the index
146                 l_this_lvl_tbl.delete ;
147                 l_lower_lvl_tbl.delete ;
148                 l_costcmpnt_ids.delete;
149                 l_skip_details := 'N' ;
150                 l_idx         := 0 ;
151                 l_idx1       := 0 ;
152
153                 l_header_rec.calendar_code     := Get_Field(l_line,l_delimiter,2) ;
154                 l_header_rec.period_code      := Get_Field(l_line,l_delimiter,3) ;
155                 l_header_rec.cost_mthd_code   := Get_Field(l_line,l_delimiter,4) ;
156                 l_header_rec.whse_code       := Get_Field(l_line,l_delimiter,5) ;
157                 l_header_rec.item_id         := Get_Field(l_line,l_delimiter,6) ;
158                 l_header_rec.item_no        := Get_Field(l_line,l_delimiter,7) ;
159                 l_header_rec.user_name       := Get_Field(l_line,l_delimiter,8) ;
160
161                 UTL_FILE.PUT_LINE(l_log_handle, 'Type           = ' || l_type) ;
162                 UTL_FILE.PUT_LINE(l_log_handle, 'calendar_code = ' || l_header_
rec.calendar_code) ;
163                 UTL_FILE.PUT_LINE(l_log_handle, 'period_code   = ' || l_header_
rec.period_code) ;
164                 UTL_FILE.PUT_LINE(l_log_handle, 'cost_mthd_code = ' || l_header_
rec.cost_mthd_code) ;
165                 UTL_FILE.PUT_LINE(l_log_handle, 'whse_code     = ' || l_header_
rec.whse_code) ;

```

```

166         UTL_FILE.PUT_LINE(l_log_handle, 'item_id           = ' || l_header_
rec.item_id) ;
167         UTL_FILE.PUT_LINE(l_log_handle, 'item_no           = ' || l_header_
rec.item_no) ;
168         UTL_FILE.PUT_LINE(l_log_handle, 'user_name          = ' || l_header_
rec.user_name) ;
169
170         /* This Level Detail record. Skip details in case errors in loading
header record */
171         ELSIF l_type = 20 AND l_skip_details = 'Y' THEN
172             UTL_FILE.PUT_LINE(l_log_handle, 'Error : Skipping this record..');
173             ELSIF l_type = 20 AND l_skip_details = 'N' THEN
174                 l_idx := l_idx + 1 ;
175                 l_this_lvl_tbl(l_idx).cmpntcost_id           := Get_Field(l_line,l_
delimiter,2) ;
176                 l_this_lvl_tbl(l_idx).cost_cmpntcls_id       := Get_Field(l_line,l_
delimiter,3) ;
177                 l_this_lvl_tbl(l_idx).cost_cmpntcls_code     := Get_Field(l_line,l_
delimiter,4) ;
178                 l_this_lvl_tbl(l_idx).cost_analysis_code     := Get_Field(l_line,l_
delimiter,5) ;
179                 l_this_lvl_tbl(l_idx).cmpnt_cost             := Get_Field(l_line,l_
delimiter,6) ;
180                 l_this_lvl_tbl(l_idx).burden_ind             := Get_Field(l_line,l_
delimiter,7) ;
181                 l_this_lvl_tbl(l_idx).total_qty              := Get_Field(l_line,l_
delimiter,8) ;
182                 l_this_lvl_tbl(l_idx).costcalc_orig          := Get_Field(l_line,l_
delimiter,9) ;
183                 l_this_lvl_tbl(l_idx).rmcalc_type            := Get_Field(l_line,l_
delimiter,10) ;
184                 l_this_lvl_tbl(l_idx).delete_mark           := Get_Field(l_line,l_
delimiter,11) ;
185                 l_this_lvl_tbl(l_idx).attribute1             := Get_Field(l_line,l_
delimiter,12) ;
186                 l_this_lvl_tbl(l_idx).attribute2             := Get_Field(l_line,l_
delimiter,13) ;
187                 l_this_lvl_tbl(l_idx).attribute3             := Get_Field(l_line,l_
delimiter,14) ;
188                 l_this_lvl_tbl(l_idx).attribute4             := Get_Field(l_line,l_
delimiter,15) ;
189                 l_this_lvl_tbl(l_idx).attribute5             := Get_Field(l_line,l_
delimiter,16) ;
190                 l_this_lvl_tbl(l_idx).attribute6             := Get_Field(l_line,l_
delimiter,17) ;

```

```
191      l_this_lvl_tbl(l_idx).attribute7      := Get_Field(l_line,l_
delimiter,18) ;
192      l_this_lvl_tbl(l_idx).attribute8      := Get_Field(l_line,l_
delimiter,19) ;
193      l_this_lvl_tbl(l_idx).attribute9      := Get_Field(l_line,l_
delimiter,20) ;
194      l_this_lvl_tbl(l_idx).attribute10     := Get_Field(l_line,l_
delimiter,21) ;
195      l_this_lvl_tbl(l_idx).attribute11     := Get_Field(l_line,l_
delimiter,22) ;
196      l_this_lvl_tbl(l_idx).attribute12     := Get_Field(l_line,l_
delimiter,23) ;
197      l_this_lvl_tbl(l_idx).attribute13     := Get_Field(l_line,l_
delimiter,24) ;
198      l_this_lvl_tbl(l_idx).attribute14     := Get_Field(l_line,l_
delimiter,25) ;
199      l_this_lvl_tbl(l_idx).attribute15     := Get_Field(l_line,l_
delimiter,26) ;
200      l_this_lvl_tbl(l_idx).attribute16     := Get_Field(l_line,l_
delimiter,27) ;
201      l_this_lvl_tbl(l_idx).attribute17     := Get_Field(l_line,l_
delimiter,28) ;
202      l_this_lvl_tbl(l_idx).attribute18     := Get_Field(l_line,l_
delimiter,29) ;
203      l_this_lvl_tbl(l_idx).attribute19     := Get_Field(l_line,l_
delimiter,30) ;
204      l_this_lvl_tbl(l_idx).attribute20     := Get_Field(l_line,l_
delimiter,31) ;
205      l_this_lvl_tbl(l_idx).attribute21     := Get_Field(l_line,l_
delimiter,32) ;
206      l_this_lvl_tbl(l_idx).attribute22     := Get_Field(l_line,l_
delimiter,33) ;
207      l_this_lvl_tbl(l_idx).attribute23     := Get_Field(l_line,l_
delimiter,34) ;
208      l_this_lvl_tbl(l_idx).attribute24     := Get_Field(l_line,l_
delimiter,35) ;
209      l_this_lvl_tbl(l_idx).attribute25     := Get_Field(l_line,l_
delimiter,36) ;
210      l_this_lvl_tbl(l_idx).attribute26     := Get_Field(l_line,l_
delimiter,37) ;
211      l_this_lvl_tbl(l_idx).attribute27     := Get_Field(l_line,l_
delimiter,38) ;
212      l_this_lvl_tbl(l_idx).attribute28     := Get_Field(l_line,l_
delimiter,39) ;
213      l_this_lvl_tbl(l_idx).attribute29     := Get_Field(l_line,l_
```

```

delimiter,40) ;
214     l_this_lvl_tbl(l_idx).attribute30           := Get_Field(l_line,l_
delimiter,41) ;
215     l_this_lvl_tbl(l_idx).attribute_category   := Get_Field(l_line,l_
delimiter,42) ;
216
217     UTL_FILE.PUT_LINE(l_log_handle,'cmpntcost_id('||l_idx||')      = '
||
218     l_this_lvl_tbl(l_idx).cmpntcost_id) ;
219     UTL_FILE.PUT_LINE(l_log_handle,'cost_cmpntcls_id('||l_idx||')  = '
||
220     l_this_lvl_tbl(l_idx).cost_cmpntcls_id) ;
221     UTL_FILE.PUT_LINE(l_log_handle,'cost_cmpntcls_code('||l_idx||') = '||
222     l_this_lvl_tbl(l_idx).cost_cmpntcls_code) ;
223     UTL_FILE.PUT_LINE(l_log_handle,'cost_analysis_code('||l_idx||') = '||
224     l_this_lvl_tbl(l_idx).cost_analysis_code) ;
225     UTL_FILE.PUT_LINE(l_log_handle,'cmpnt_cost('||l_idx||')      = '
||
226     l_this_lvl_tbl(l_idx).cmpnt_cost) ;
227     UTL_FILE.PUT_LINE(l_log_handle,'burden_ind('||l_idx||')      = '
||
228     l_this_lvl_tbl(l_idx).burden_ind) ;
229     UTL_FILE.PUT_LINE(l_log_handle,'total_qty('||l_idx||')      = '
||
230     l_this_lvl_tbl(l_idx).total_qty) ;
231     UTL_FILE.PUT_LINE(l_log_handle,'costcalc_orig('||l_idx||')    = '
||
232     l_this_lvl_tbl(l_idx).costcalc_orig) ;
233     UTL_FILE.PUT_LINE(l_log_handle,'rmcalc_type('||l_idx||')     = '
||
234     l_this_lvl_tbl(l_idx).rmcalc_type) ;
235     UTL_FILE.PUT_LINE(l_log_handle,'delete_mark('||l_idx||')     = '
||
236     l_this_lvl_tbl(l_idx).delete_mark) ;
237
238     /* Lower Level Detail record. Skip details in case errors in loading
header record */
239     ELSIF l_type = 30 AND l_skip_details = 'Y' THEN
240         UTL_FILE.PUT_LINE(l_log_handle, 'Error : Skipping this record...');
241     ELSIF l_type = 30 AND l_skip_details = 'N' THEN
242         l_idx1 := l_idx1 + 1 ;
243         l_type           := Get_Field(l_line,l_
delimiter,1) ;
244         l_lower_lvl_tbl(l_idx1).cmpntcost_id       := Get_Field(l_line,l_
delimiter,2) ;

```

```

245         l_lower_lvl_tbl(l_idx1).cost_cmpntcls_id      := Get_Field(l_line,l_
delimiter,3) ;
246         l_lower_lvl_tbl(l_idx1).cost_cmpntcls_code   := Get_Field(l_line,l_
delimiter,4) ;
247         l_lower_lvl_tbl(l_idx1).cost_analysis_code   := Get_Field(l_line,l_
delimiter,5) ;
248         l_lower_lvl_tbl(l_idx1).cmpnt_cost           := Get_Field(l_line,l_
delimiter,6) ;
249
250         UTL_FILE.PUT_LINE(l_log_handle,'ll cmpntcost_id('||l_idx1||')      =
' ||
251         l_lower_lvl_tbl(l_idx1).cmpntcost_id) ;
252         UTL_FILE.PUT_LINE(l_log_handle,'ll cost_cmpntcls_id('||l_idx1||') =
' ||
253         l_lower_lvl_tbl(l_idx1).cost_cmpntcls_id)      ;
254         UTL_FILE.PUT_LINE(l_log_handle,'ll cost_cmpntcls_code('||l_idx1||') =
' ||
255         l_lower_lvl_tbl(l_idx1).cost_cmpntcls_code) ;
256         UTL_FILE.PUT_LINE(l_log_handle,'ll cost_analysis_code('||l_idx1||') =
' ||
257         l_lower_lvl_tbl(l_idx1).cost_analysis_code) ;
258         UTL_FILE.PUT_LINE(l_log_handle,'ll cmpnt_cost('||l_idx1||')      =
' ||
259         l_lower_lvl_tbl(l_idx1).cmpnt_cost) ;
260
261         END IF ;
262
263     EXCEPTION
264     WHEN OTHERS THEN
265         /* in case of any errors log the error and continue with the next
record
266         * in case of header, skip all the detail record
267         * in case details, skip that particular record */
268
269         UTL_FILE.PUT_LINE(l_outfile_handle, 'Error : ' || to_char(SQLCODE) || '
' || SQLERRM);
270         UTL_FILE.PUT_LINE(l_log_handle, 'Error : ' || to_char(SQLCODE) || '
' || SQLERRM);
271
272         IF l_type = 10 THEN
273             l_skip_details := 'Y' ;
274             UTL_FILE.PUT_LINE(l_log_handle, 'Error : Skip detail records.');
```

```

278         l_lower_lvl_tbl.delete(l_idx1);
279     END IF ;
280
281     END ;
282
283     BEGIN
284
285         UTL_FILE.GET_LINE(l_infile_handle, l_line);
286         l_record_count :=l_record_count+1;
287         UTL_FILE.NEW_LINE(l_log_handle);
288         l_type := Get_Field(l_line,l_delimiter,1) ; -- 10 : header rec, 20
: this level, 30 : lower level
289
290     EXCEPTION
291         /* End of file */
292         WHEN NO_DATA_FOUND THEN
293             IF l_skip_details = 'N' THEN
294
295                 GMF_ItemCost_PUB.Create_Item_Cost
296                 ( p_api_version      => 1.0
297                 , p_init_msg_list    => FND_API.G_TRUE
298                 , p_commit          => FND_API.G_TRUE
299
300                 , x_return_status    => l_status
301                 , x_msg_count        => l_count
302                 , x_msg_data         => l_data
303
304                 , p_header_rec       => l_header_rec
305                 , p_this_level_dtl_tbl => l_this_lvl_tbl
306                 , p_lower_level_dtl_Tbl => l_lower_lvl_tbl
307
308                 , x_costcmpnt_ids    => l_costcmpnt_ids
309                 );
310                 l_continue := 'N' ;
311                 goto GET_MSG_STACK ;
312             END IF ;
313     END;
314
315     IF (l_type = 10 AND l_record_count != 1 AND l_skip_details = 'N') THEN
316
317         DBMS_OUTPUT.PUT_LINE('Calling Create_Item_Cost API...');
318
319         GMF_ItemCost_PUB.Create_Item_Cost
320         ( p_api_version      => 1.0
321         , p_init_msg_list    => FND_API.G_TRUE

```

```
322     , p_commit           => FND_API.G_TRUE
323
324     , x_return_status    => l_status
325     , x_msg_count       => l_count
326     , x_msg_data        => l_data
327
328     , p_header_rec      => l_header_rec
329     , p_this_level_dtl_tbl => l_this_lvl_tbl
330     , p_lower_level_dtl_Tbl => l_lower_lvl_tbl
331
332     , x_costcmpnt_ids    => l_costcmpnt_ids
333   );
334   DBMS_OUTPUT.PUT_LINE('after API call. status := ' || l_status ||' cnt
:= ' || l_count );
335   END IF;
336
337   <<GET_MSG_STACK>>
338   NULL;
339
340   /* Check if any messages generated. If so then decode and */
341   /* output to error message flat file */
342
343   IF l_count > 0 THEN
344
345     l_loop_cnt :=1;
346     LOOP
347       FND_MSG_PUB.Get(
348         p_msg_index    => l_loop_cnt,
349         p_data         => l_data,
350         p_encoded      => FND_API.G_FALSE,
351         p_msg_index_out => l_dummy_cnt);
352
353       --DBMS_OUTPUT.PUT_LINE(l_data );
354       UTL_FILE.PUT_LINE(l_log_handle, l_data);
355
356       /* Update error status */
357       IF (l_status = 'U') THEN
358         l_return_status :=l_status;
359       ELSIF (l_status = 'E' and l_return_status <> 'U') THEN
360         l_return_status :=l_status;
361       ELSE
362         l_return_status :=l_status;
363       END IF;
364
365       l_loop_cnt := l_loop_cnt + 1;
```

```

366         IF l_loop_cnt > l_count THEN
367             EXIT;
368         END IF;
369
370     END LOOP; -- msg stack loop
371     l_count := 0 ;
372
373     END IF;-- if count of msg stack > 0
374
375     DBMS_OUTPUT.PUT_LINE('# of CostIds inserted : ' || l_costcmpnt_
ids.count);
376     FOR i in 1..l_costcmpnt_ids.count
377     LOOP
378         UTL_FILE.PUT_LINE(l_log_handle,
379             'CmpntClsId : '      || l_costcmpnt_ids(i).cost_cmpntcls_id ||
380             ' Analysis Code : ' || l_costcmpnt_ids(i).cost_analysis_code ||
381             ' Cost Level : '    || l_costcmpnt_ids(i).cost_level ||
382             ' CostId : '       || l_costcmpnt_ids(i).cmpntcost_id);
383     END LOOP ;
384
385     IF l_continue = 'N' THEN
386         EXIT ;
387     END IF ;
388
389     END LOOP;
390
391     UTL_FILE.NEW_LINE(l_log_handle);
392     UTL_FILE.PUT_LINE(l_log_handle, 'Process Completed at ' || to_
char(SYSDATE,'DD-MON-YY HH24:MI:SS'));
393     UTL_FILE.FCLOSE_ALL;
394
395     RETURN l_return_status;
396
397     EXCEPTION
398     WHEN UTL_FILE.INVALID_OPERATION THEN
399         /* DBMS_OUTPUT.PUT_LINE('Invalid Operation For ' || l_global_file); */
400         UTL_FILE.FCLOSE_ALL;
401         RETURN l_return_status;
402
403     WHEN UTL_FILE.INVALID_PATH THEN
404         /* DBMS_OUTPUT.PUT_LINE('Invalid Path For      ' || l_global_file); */
405         UTL_FILE.FCLOSE_ALL;
406         RETURN l_return_status;
407
408     WHEN UTL_FILE.INVALID_MODE THEN

```

```

409      /* DBMS_OUTPUT.PUT_LINE('Invalid Mode For      ' || l_global_file); */
410      UTL_FILE.FCLOSE_ALL;
411      RETURN l_return_status;
412
413  WHEN UTL_FILE.INVALID_FILEHANDLE THEN
414      /* DBMS_OUTPUT.PUT_LINE('Invalid File Handle  ' || l_global_file); */
415      UTL_FILE.FCLOSE_ALL;
416      RETURN l_return_status;
417
418  WHEN UTL_FILE.WRITE_ERROR THEN
419      /* DBMS_OUTPUT.PUT_LINE('Invalid Write Error  ' || l_global_file); */
420      UTL_FILE.FCLOSE_ALL;
421      RETURN l_return_status;
422
423  WHEN UTL_FILE.READ_ERROR THEN
424      /* DBMS_OUTPUT.PUT_LINE('Invalid Read  Error  ' || l_global_file); */
425      UTL_FILE.FCLOSE_ALL;
426      RETURN l_return_status;
427
428  WHEN UTL_FILE.INTERNAL_ERROR THEN
429      /* DBMS_OUTPUT.PUT_LINE('Internal Error'); */
430      UTL_FILE.FCLOSE_ALL;
431      RETURN l_return_status;
432
433  WHEN OTHERS THEN
434      /* DBMS_OUTPUT.PUT_LINE('Other Error'); */
435      UTL_FILE.PUT_LINE(l_outfile_handle, 'Error : ' || to_char(SQLCODE) || ' '
|| SQLERRM);
436      UTL_FILE.PUT_LINE(l_log_handle, 'Error : ' || to_char(SQLCODE) || ' ' ||
SQLERRM);
437      UTL_FILE.PUT_LINE(l_log_handle, 'Process Completed at ' || to_
char(SYSDATE, 'DD-MON-YY HH24:MI:SS'));
438      UTL_FILE.FCLOSE_ALL;
439      l_return_status := 'U' ;
440      RETURN l_return_status;
441
442  END Create_Item_Cost;
443
444
445
--+=====+
446  --| FUNCTION NAME
|
447  --|   Get_Field
|

```

```

448 --|
|
449 --| TYPE
|
450 --|   Public
|
451 --|
|
452 --| USAGE
|
453 --|   Get value of field n from a delimited line of ASCII data
|
454 --|
|
455 --| DESCRIPTION
|
456 --|   This utility function will return the value of a field from
|
457 --|   a delimited line of ASCII text
|
458 --|
|
459 --| PARAMETERS
|
460 --|   p_line           IN VARCHAR2       - line of data
|
461 --|   p_delimiter      IN VARCHAR2       - Delimiter character
|
462 --|   p_field_no      IN NUMBER         - Field occurrence to be
|
463 --|                                     returned
|
464 --|
|
465 --| RETURNS
|
466 --|   VARCHAR2        - Value of field
|
467 --|
|
468 --| HISTORY
|
469 --|
|
470

```

```
--+=====+
471 -- Api end of comments
472
473 FUNCTION Get_Field
474 ( p_line          IN VARCHAR2
475 , p_delimiter    IN VARCHAR2
476 , p_field_no     IN NUMBER
477 )
478 RETURN VARCHAR2
479 IS
480
481 /* Local variables */
482 l_start          NUMBER :=0;
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     THEN
495         RETURN NULL;
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 THEN
503     l_end        := LENGTH(p_line) + 1;
504 END IF;
505
506 /* Extract the field data */
507 IF (l_end - l_start) = 1
508 THEN
509     RETURN NULL;
510 ELSE
511     RETURN SUBSTR(p_line,(l_start + 1),((l_end - l_start) - 1));
512 END IF;
513
514 EXCEPTION
```

```

515     WHEN OTHERS
516     THEN
517         RETURN NULL;
518
519 END Get_Field;
520
521
-- +-----+
522 --| FUNCTION NAME
|
523 --|     Get_Substring
|
524 --|
|
525 --| TYPE
|
526 --|     Public
|
527 --|
|
528 --| USAGE
|
529 --|     Get value of Sub-string from formatted ASCII data file record
|
530 --|
|
531 --| DESCRIPTION
|
532 --|     This utility function will return the value of a passed sub-string
|
533 --|     of a formatted ASCII data file record
|
534 --|
|
535 --| PARAMETERS
|
536 --|     p_substring           IN VARCHAR2           - substring data
|
537 --|
|
538 --| RETURNS
|
539 --|     VARCHAR2               - Value of field
|
540 --|

```

```
|
541 --| HISTORY
|
542 --|
|
543
--+=-----+
544 -- Api end of comments
545
546 FUNCTION Get_Substring
547 ( p_substring    IN VARCHAR2
548 )
549 RETURN VARCHAR2
550 IS
551
552 /* Local variables */
553 l_string_value  VARCHAR2(200) := ' ';
554
555 BEGIN
556
557 /* Determine start position */
558 l_string_value :=NVL(RTRIM(LTRIM(p_substring)),' ');
559
560 RETURN l_string_value;
561 EXCEPTION
562   WHEN OTHERS
563   THEN
564     RETURN ' ';
565
566 END Get_Substring;
```

Technical Overview

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 that particular row will be skipped and the process will continue with the next record. 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

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

Object Type	Name
Procedure - Get Item Cost	Get_Item_Cost

Resource Cost

Each function on 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

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

Allocation Definition

Each function on 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

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

Burden Details

Each function on 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 that particular row will be skipped and the process will continue with the next record. 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

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

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`

Parameter	Type	IN/OUT	Required	Validation
<code>p_api_version</code>	<code>varchar2</code>	IN	Y	Validates version compatibility. The version sent by the calling function is compared to the internal version of the API and an unexpected error (U) is generated if these do not match.
<code>p_init_msg_list</code>	<code>varchar2</code>	IN	N	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 <code>FND_API.G_FALSE</code> which means that the API will not initialize the message list.

Parameter	Type	IN/OUT	Required	Validation
p_commit	varchar2	IN	N	Used to specify whether the API should commit its work before returning to the calling function. If not supplied it will default to FND_API.G_FALSE.
x_return_status	varchar2	OUT	-	Specifies whether the API was successful or failed: 'S' Successful 'E' failed due to expected error 'U' failed due to unexpected error
x_msg_count	number	OUT	-	Specifies number of messages added to message list.
x_msg_data	varchar2	OUT	-	Returns the messages in an encoded format. These messages can then be processed by the standard message functions as defined in Business Object API Coding Standards Document.

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.

Business Objects

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:

- n Calendar
- n Period
- n Cost Method
- n Warehouse
- n Item

Following is the definition of Header_Rec_Type:

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

This Level Cost Detail Entity

“This Level Cost Detail” entity specifies exact cost figure of a given cost component and analysis code. Along with that it also specifies other details which is 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:

Attribute	Required	Description
cmpntcost_id	N	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.
cost_cmpntcls_id	N	Valid Cost Component Class ID (Either cost_cmpntcls_id or cost_cmpntcls_code is required). Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpnt_mst.
cost_cmpntcls_code	N	Valid Cost Component Class Code (Either cost_cmpntcls_id or cost_cmpntcls_code is required). Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpnt_mst.
cost_analysis_code	Y	Valid Cost Analysis Code. Cost Analysis Code must exist in cm_alys_mst.
cmpnt_cost	Y	Component Cost must be a valid number.

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

Lower Level Cost Detail Entity

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

- Cost Component Class
- Analysis Code
- Component Cost

Following is the definition of Lower_Level_Dtl_Tbl_Type:

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

Parameter - x_costcmpnt_ids (OUT)

This 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:

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

Structure and Logic

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

Package Specification - GMFPCSTS.pls

This file holds definition of all the variables, procedures and functions that will be available to users for use.

```

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

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

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                          := FND_API.G_MISS_NUM
, cost_cmpntcls_id NUMBER                          := FND_API.G_MISS_NUM
, cost_cmpntcls_code cm_cmpt_mst.cost_cmpntcls_code%TYPE:=FND_API.G_MISS_CHAR
, cost_analysis_code cm_cmpt_dtl.cost_analysis_code%TYPE:=FND_API.G_MISS_CHAR
, cmpnt_cost       NUMBER                          := FND_API.G_MISS_NUM
, delete_mark     NUMBER                          := FND_API.G_MISS_NUM
);

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                          := FND_API.G_MISS_NUM
, cost_analysis_code cm_cmpt_dtl.cost_analysis_code%TYPE:=FND_API.G_MISS_CHAR

```

```

, cost_level          NUMBER          := FND_API.G_MISS_NUM
, cmpntcost_id       NUMBER          := FND_API.G_MISS_NUM
);

TYPE costcmpnt_ids_tbl_type IS TABLE OF costcmpnt_ids_rec_type
INDEX BY BINARY_INTEGER;

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

    x_return_statusOUT VARCHAR2,
    x_msg_countOUT   VARCHAR2,
    x_msg_dataOUT   VARCHAR2,

    p_header_recIN   Header_Rec_Type,
    p_this_level_dtl_tblIN   This_Level_Dtl_Tbl_Type,
    p_lower_level_dtl_TblIN   Lower_Level_Dtl_Tbl_Type,

    x_costcmpnt_idsOUT costcmpnt_ids_tbl_type
);

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

    x_return_statusOUT VARCHAR2,
    x_msg_countOUT   VARCHAR2,
    x_msg_dataOUT   VARCHAR2,

    p_header_recIN   Header_Rec_Type,
    p_this_level_dtl_tblIN   This_Level_Dtl_Tbl_Type,
    p_lower_level_dtl_TblIN   Lower_Level_Dtl_Tbl_Type
);

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

    x_return_statusOUT VARCHAR2,
    x_msg_countOUT   VARCHAR2,
    x_msg_dataOUT   VARCHAR2,

    p_header_recIN   Header_Rec_Type,
    p_this_level_dtl_tblIN   This_Level_Dtl_Tbl_Type,

```

```
p_lower_level_dtl_tblIN Lower_Level_Dtl_Tbl_Type

);

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

    x_return_statusOUT VARCHAR2,
    x_msg_countOUT VARCHAR2,
    x_msg_dataOUT VARCHAR2,

    p_header_recIN Header_Rec_Type,

    x_this_level_dtl_tblOUT This_Level_Dtl_Tbl_Type,
    x_lower_level_dtl_tblOUT Lower_Level_Dtl_Tbl_Type

);
```

Public Procedure Create_Item_Cost

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

p_header_rec Header level Information for Item Cost record

p_this_level_dtl_tbl This level cost details

p_lower_level_dtl_tbl Lower level cost details

Note: For details on 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.

Note: During validation process, if a detail record fails to go through all the validations then only that particular detail record will be skipped/eliminated and process will continue with the next detail record.

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.

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 only that particular detail record will be skipped/eliminated and process will continue with the next detail record.

If the record is not found using either `cmpntcost_id` or using unique key then all the records for that item 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 for an Item.

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

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 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: During validation process, if a detail record fails to go through all the validations then only that particular detail record will be skipped/eliminated and process will continue with the next detail record.

If the record is not found using either `cmpntcost_id` or using unique key then all the records for that item 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.

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

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:

Attribute	Required	Description
resources	Y	Valid Resource Code must exist in cr_rsrc_mst
orgn_code	Y	Valid Organization Code must exist in sy_orgn_mst
calendar_code	Y	Valid Calendar Code must exist in cm_cldr_mst.
period_code	Y	Valid Period code must exist in cm_cldr_dtl and must not be closed.
cost_mthd_code	Y	Valid Cost Method Code must exist in cm_mthd_mst
usage_um	Y	Valid Unit of Measure must exist in sy_uoms_mst
nominal_cost	Y	Valid Number must be a number > 0
delete_mark	N	Either 0 or 1. (Default is 0)

Attribute	Required	Description
user_name	Y	Valid Oracle Application's user name. User name must exist in fnd_user

Structure and Logic

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

Package Specification - GMFPRESS.pls

This file holds definition of all the variables, procedures and functions that will be available to users for use.

```

TYPE Resource_Cost_Rec_Type IS RECORD
(
  resources cm_rsrc_dtl.resources%TYPE           := FND_API.G_MISS_CHAR ,
  orgn_code cm_rsrc_dtl.orgn_code%TYPE           := FND_API.G_MISS_CHAR ,
  calendar_code cm_rsrc_dtl.calendar_code%TYPE:= FND_API.G_MISS_CHAR ,
  period_code cm_rsrc_dtl.period_code%TYPE      := FND_API.G_MISS_CHAR ,
  cost_mthd_code cm_rsrc_dtl.cost_mthd_code%TYPE:= FND_API.G_MISS_CHAR ,
  usage_um cm_rsrc_dtl.usage_um%TYPE           := FND_API.G_MISS_CHAR ,
  nominal_cost NUMBER                           := FND_API.G_MISS_NUM ,
  delete_mark cm_rsrc_dtl.delete_mark%TYPE      := 0 ,
  user_name fnd_user.user_name%TYPE            := FND_API.G_MISS_CHAR
);

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

  x_return_statusOUT VARCHAR2,
  x_msg_countOUT VARCHAR2,
  x_msg_dataOUT VARCHAR2,

  p_resource_cost_recIN Resource_Cost_Rec_Type
);

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

  x_return_statusOUT VARCHAR2,
  x_msg_countOUT VARCHAR2,

```

```
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN Resource_Cost_Rec_Type
);

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

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN Resource_Cost_Rec_Type
);

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

x_return_statusOUT VARCHAR2,
x_msg_countOUT VARCHAR2,
x_msg_dataOUT VARCHAR2,

p_resource_cost_recIN Resource_Cost_Rec_Type,
x_resource_cost_recOUT Resource_Cost_Rec_Type
);
```

Public Procedure Create_Resource_Cost

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

p_resource_cost_rec Information for Resource Cost record

Note: For details on 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.

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

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

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

x_resource_cost_rec Information for Resource Cost record

Allocation Definition

Allocation Definition specifies information for an allocation code about the items to which expenses 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:

Attribute	Required	Description
alloc_id	Y	Surrogate for Allocation Code (Either alloc_id OR co_code + alloc_code is required.)
co_code	Y	Valid Company Code (Either alloc_id OR co_code + alloc_code is required.)
alloc_code	Y	Valid Allocation Code (Either alloc_id OR co_code + alloc_code is required.)
alloc_method	Y	Valid Allocation Method (Either 0 or 1)

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

Structure and Logic

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

Package Specification - GMFPALCS.pls

This file holds definition of all the variables, procedures and functions that will be available to users for use.

```

TYPE Allocation_Definition_Rec_Type IS RECORD
(
  alloc_id          NUMBER                := FND_API.G_MISS_NUM ,
  alloc_code        gl_alloc_mst.alloc_code%TYPE := FND_API.G_MISS_CHAR ,
  co_code           sy_orgn_mst.co_code%TYPE   := FND_API.G_MISS_CHAR ,
  alloc_method      NUMBER                := FND_API.G_MISS_NUM ,
  line_no           NUMBER                := FND_API.G_MISS_NUM ,
  item_id           NUMBER                := FND_API.G_MISS_NUM ,
  item_no           ic_item_mst.item_no%TYPE  := FND_API.G_MISS_CHAR ,
  basis_account_key gl_alloc_bas.basis_account_key%TYPE:= FND_API.G_MISS_CHAR ,
  balance_type      NUMBER                := FND_API.G_MISS_NUM ,
  bas_ytd_ptd       NUMBER                := FND_API.G_MISS_NUM ,
  fixed_percent     NUMBER                := FND_API.G_MISS_NUM ,
  cmpntcls_id       NUMBER                := FND_API.G_MISS_NUM ,
  cost_cmpntcls_code cm_cmpt_mst.cost_cmpntcls_code%TYPE := FND_API.G_MISS_CHAR ,
  analysis_code     cm_alys_mst.cost_analysis_code%TYPE := FND_API.G_MISS_CHAR ,
  whse_code         gl_alloc_bas.whse_code%TYPE := FND_API.G_MISS_CHAR ,
  delete_mark       gl_alloc_bas.delete_mark%TYPE := 0 ,
  user_name         fnd_user.user_name%TYPE := FND_API.G_MISS_CHAR
);

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

  x_return_status OUT VARCHAR2,
  x_msg_count OUT VARCHAR2,
  x_msg_data OUT VARCHAR2,

  p_allocation_definition_rec IN Allocation_Definition_Rec_Type
);

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

  x_return_status OUT VARCHAR2,
  x_msg_count OUT VARCHAR2,

```

```

x_msg_dataOUT VARCHAR2,

p_allocation_definition_rec      IN Allocation_Definition_Rec_Type
);

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

    x_return_statusOUT VARCHAR2,
    x_msg_countOUT VARCHAR2,
    x_msg_dataOUT VARCHAR2,

    p_allocation_definition_rec      IN Allocation_Definition_Rec_Type
);

```

Public Procedure Create_Allocation_Definition

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

p_allocation_definition_rec Information for Allocation Definition record

Note: For details on 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

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:

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

Following is the definition of Header_Rec_Type:

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

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:

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

Following is the definition of This_Level_Dtl_Tbl_Type:

Attribute	Required	Description
burdenline_id	N	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.
resources	Y	Valid Resource Code must exist in cr_rsrc_mst
cost_cmpntcls_id	N	Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst and must have usage indicator = 2 (Burden/Overhead).
cost_cmpntcls_code	N	Cost Component Class ID/ Cost Component Class Code must exist in cm_cmpt_mst and must have usage indicator = 2 (Burden/Overhead).
cost_analysis_code	Y	Valid Cost Analysis Code must exist in cm_alys_mst.
burden_usage	Y	Valid Number between 0.00000001 and 999999999.99999999

Attribute	Required	Description
item_qty	Y	Valid Number between 0.000000001 and 999999999.999999999
item_um	Y	Item UOM must be a valid Unit of Measure. Since this Item UOM can be any UOM, conversion factor should exist for this UOM and items actual UOM.
burden_qty	Y	Valid Number between 0.000000001 and 999999999.999999999
burden_um	Y	Burden UOM must be a valid Unit of Measure and Burden UOM type should be same as Resource UOM Type.
Delete_mark	Y	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.

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:

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

Structure and Logic

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

Package Specification - GMFPBRDS.pls

This file holds definition of all the variables, procedures and functions that will be available to users for use.

```

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

TYPE Burden_Dtl_Rec_Type IS RECORD
(
  burdenline_id  NUMBER                          := FND_API.G_MISS_NUM  ,
  resources      cr_rsrc_mst.resources%TYPE      := FND_API.G_MISS_CHAR ,
  cost_cmpntcls_id NUMBER                        := FND_API.G_MISS_NUM  ,
  cost_cmpntcls_codecm_cmpt_mst.cost_cmpntcls_code%TYPE:=FND_API.G_MISS_CHAR,
  cost_analysis_code cm_alys_mst.cost_analysis_code%TYPE:=FND_API.G_MISS_
  CHAR,
  burden_usage   NUMBER                          := FND_API.G_MISS_NUM  ,
  item_qty       NUMBER                          := FND_API.G_MISS_NUM  ,
  item_um        cm_brnd_dtl.item_um%TYPE        := FND_API.G_MISS_CHAR ,
  burden_qty     NUMBER                          := FND_API.G_MISS_NUM  ,
  burden_um      cm_brnd_dtl.item_um%TYPE        := FND_API.G_MISS_CHAR ,
  burden_factor  NUMBER                          := FND_API.G_MISS_NUM  ,
  delete_mark    cm_brnd_dtl.delete_mark%TYPE    := 0
);

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      := FND_API.G_MISS_CHAR ,
  cost_cmpntcls_id NUMBER                        := FND_API.G_MISS_NUM  ,
  cost_analysis_code cm_brnd_dtl.cost_analysis_code%TYPE:=FND_API.G_MISS_
  CHAR,
  burdenline_id  NUMBER                          := FND_API.G_MISS_NUM
);

```

```
TYPE Burdenline_Ids_Tbl_Type IS TABLE OF Burdenline_Ids_Rec_Type
    INDEX BY BINARY_INTEGER;
PROCEDURE Create_Burden_Details
(
    p_api_version IN NUMBER ,
    p_init_msg_list IN VARCHAR2 := FND_API.G_FALSE,
    p_commit IN VARCHAR2 := FND_API.G_FALSE,

    x_return_status OUT VARCHAR2,
    x_msg_count OUT VARCHAR2,
    x_msg_data OUT VARCHAR2,

    p_header_rec IN Burden_Header_Rec_Type,
    p_dtl_tbl IN Burden_Dtl_Tbl_Type,

    x_burdenline_ids IN Burdenline_Ids_Tbl_Type
);

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

    x_return_status OUT VARCHAR2,
    x_msg_count OUT VARCHAR2,
    x_msg_data OUT VARCHAR2,

    p_header_rec IN Burden_Header_Rec_Type,
    p_dtl_tbl IN Burden_Dtl_Tbl_Type,
);

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

    x_return_status OUT VARCHAR2,
    x_msg_count OUT VARCHAR2,
    x_msg_data OUT VARCHAR2,

    p_header_rec IN Burden_Header_Rec_Type,
    p_dtl_tbl IN Burden_Dtl_Tbl_Type,
);

PROCEDURE Get_Item_Cost
(
    p_api_version IN NUMBER ,
```

```
p_init_msg_list          IN  VARCHAR2 := FND_API.G_FALSE,  
  
x_return_statusOUT      VARCHAR2,  
x_msg_countOUT          VARCHAR2,  
x_msg_dataOUT           VARCHAR2,  
  
p_header_recIN          Burden_Header_Rec_Type,  
  
x_dtl_tblOUT            Burden_Dtl_Tbl_Type  
);
```

Public Procedure Create_Burden_Details

Procedure Create_Burden_Details is used to insert standard resource burdens. API caller 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.

Note: During validation process, if a detail record fails to go through all the validations then only that particular detail record will be skipped/eliminated and process will continue with the next detail record.

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.

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 only that particular detail record will be skipped/eliminated and process will continue with the next detail record.

If the record is not found using either `cmpntcost_id` or using unique key then all the records for that item 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.

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 only that particular detail record will be skipped/eliminated and process will continue with the next detail record.

If the record is not found using either `cmpntcost_id` or using unique key then all the records for that item 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.

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

Messages and Errors

This appendix covers:

- Handling Messages
- Interpreting Error Conditions
- Understanding Error Messages

Handling Messages

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

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

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

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

Initialize Initializes the API message list.

Add Adds a message to the API message list.

Get Gets a message from the API message list.

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

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

Reset Resets the index used in getting messages.

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

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

To add a message to the API message list, developers 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:

- n message count
- n message data

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

```
x_msg_count          OUT          NUMBER
x_msg_dataOUTVARCHAR2
```

Example:

```

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

x_return_statusOUTVARCHAR2 ,
x_msg_countOUTNUMBER,
x_msg_dataOUTVARCHAR2 ,

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

Validate_Line
(l_return_status,
l_line_rec
);

Price_Line
(l_return_status,
l_line_rec
);

Insert_Line

```

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

Interpreting Error Conditions

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

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

Understanding Error Messages

These error messages 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 ampersand (&) is a token, or placeholder, for an actual value that will be populated at runtime.

Message Name	Message Code
Error: Burden Details delete failed for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_BRDN_DEL_FAILED_DTLS
Error: Either Allocation Id or combination of Allocation Code and Company Code is required.	GMF_API_ALLOC_DTL_REQ
Error: Allocation Method is required.	GMF_API_ALLOC_MTHD_REQ
Error: Analysis Code is required.	GMF_API_ANALYSIS_CODE_REQ
Error: Balance Type is required.	GMF_API_BALANCE_TYPE_REQ
Error: Basis Account Key is required.	GMF_API_BAS_ACC_KEY_REQ
Error: YTD/PTD indicator for balance is required.	GMF_API_BAS_YTD_PTD_REQ
Error: Burden Details delete failed for Burden line Id &BURDENLINE_ID.	GMF_API_BRDN_DEL_FAILED_ID
Error: Burden Details insert failed for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_BRDN_INSERT_FAILED
Error: No Burden Details found to update for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_BRDN_NOT_FOUND_FOR_DTL
Error: No Burden Details found to update for Burden Line Id &BURDENLINE_ID.	GMF_API_BRDN_NOT_FOUND_FOR_ID
Error: Burden Details update failed for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_BRDN_UPD_FAILED_DTLS
Error: Burden Details update failed for Burden line Id &BURDENLINE_ID.	GMF_API_BRDN_UPD_FAILED_ID
Error: Burden Indicator is required.	GMF_API_BURDEN_IND_REQ
Error: Burden Quantity is required.	GMF_API_BURDEN_QTY_REQ
Error: Burden UOM is required.	GMF_API_BURDEN_UM_REQ

Message Name	Message Code
Error: Burden Indicator supplied is &BURDEN_IND, but Usage of the Cost Component Class &CMPNT_CLS is not burden detail.	GMF_API_BURDEN_USAGE_IND
Error: Burden Usage is required.	GMF_API_BURDEN_USAGE_REQ
Error: Calendar Code is required.	GMF_API_CALENDAR_CODE_REQ
Error: Record cannot be marked for purge using Update API.	GMF_API_CANT_MARK_FOR_PURGE
Error: Period &PERIOD_CODE for Calendar &CALENDAR_CODE is Closed.	GMF_API_CLOSED_PERIOD
Error: Either Component Class Id or Component Class Code is required.	GMF_API_CMPNTCLS_ID_REQ
Error: Usage of Component Class &CMPNTCLS is not GL Expense Allocation.	GMF_API_CMPNTCLS_USG_NOT_ALC
Error: Usage of Component Class &CMPNTCLS_ID is not Burden (Overhead).	GMF_API_CMPNTCLS_USG_NOT_BRDN
Error: Component Cost is required.	GMF_API_CMPNT_COST_REQ
Error: Cost Method Code is required.	GMF_API_COST_MTHD_CODE_REQ
Error: Item Cost details not found for Item Id &ITEM_ID, Warehouse &WHSE_CODE, Calendar &CALENDAR_CODE, Period &PERIOD_CODE, Cost Method &COST_MTHD, Component Class &CMPNTCLS_ID, Analysis Code &ANALYSIS_CODE and Cost Level &COST_LEVEL.	GMF_API_COST_ROWS_NOT_FOUND
DEBUG: &MSG	GMF_API_DEBUG
Error: Delete Mark is required.	GMF_API_DELETE_MARK_REQ
Error: No Burden Details found to delete for Resource &RESOURCE, Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_DEL_BRDN_NOT_FOUND_DTL
Error: No Burden Details found to delete for Burden Line Id &BURDENLINE_ID.	GMF_API_DEL_BRDN_NOT_FOUND_ID
Error: No Component Cost Details found to delete for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_DEL_IC_NOT_FOUND_DTL

Message Name	Message Code
Error: No Component Cost Details found to delete for Component Cost Id &CMPNTCOST_ID.	GMF_API_DEL_IC_NOT_FOUND_ID
Error: Duplicate Item Cost details for Item Id &ITEM_ID, Warehouse &WHSE_CODE, Calendar &CALENDAR_CODE, Period &PERIOD_CODE, Cost Method &COST_MTHD, Component Class &CMPNTCLS_ID, Analysis Code &ANALYSIS_CODE and Cost Level &COST_LEVEL.	GMF_API_DUPLICATE_ITEM_COST
Error: Duplicate Resource Cost for Resource Code: &RESOURCES, Orgn: &ORGN_CODE, Calendar: &CALENDAR_CODE, Period: &PERIOD_CODE and Cost Method: &COST_MTHD_CODE.	GMF_API_DUPLICATE_RSRC_COST
Error: Percentage is required.	GMF_API_FIXED_PERCENT_REQ
Error: No Updates/Deletes allowed since Period &PERIOD_CODE for Calendar &CALENDAR_CODE is Frozen.	GMF_API_FROZEN_PERIOD
Error: Cannot insert new component details since Item &ITEM is frozen for Whse &WHSE, Calendar &CALENDAR, Period &PERIOD and Cost Method &COST_MTHD.	GMF_API_IC_CANNT_INSERT_CMPTS
Error: Component Cost delete failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_DEL_FAILED_DTLS
Error: Component Cost delete failed for Component Cost Id &CMPNTCOST_ID.	GMF_API_IC_DEL_FAILED_ID
Error: Lower Level Component Cost insert failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_LWRLVL_INS_FAILED
Error: This Level Component Cost insert failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_THISLVL_INS_FAILED
Error: Component Cost update failed for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_IC_UPD_FAILED_DTLS
Error: Component Cost update failed for Component Cost Id &CMPNTCOST_ID.	GMF_API_IC_UPD_FAILED_ID

Message Name	Message Code
Warning: Allocation Code &ALLOC_CODE and Company Code &CO_CODE values are ignored since allocation Id is supplied.	GMF_API_IGNORE_ALLOC_CODE
Warning: Basis Amount fields are ignored since fixed percent is used for Allocation (Allocation Method is Fixed Percent).	GMF_API_IGNORE_BASIS
Warning: BurdenLine Id &BURDENLINE_ID is supplied and will be used to Update/Delete the record. Unique key combination of Organization, Item, Warehouse, Calendar, Period, Cost Method, Resource, Component Class and Analysis Code will be ignored.	GMF_API_IGNORE_BRDN_UNIQUE_KEY
Warning: Component Class Code &CMPNTCLS_CODE is ignored since Component Class Id is supplied.	GMF_API_IGNORE_CMPNTCLS_CODE
Warning: Fixed Percent &FIXED_PERCENT value is ignored since Alloc Percent Depends on Basis Amount (Allocation Method is Basis Type).	GMF_API_IGNORE_FIXED_PERCENT
Warning: Component Cost Id &CMPNTCOST_ID is supplied and will be used to Update/Delete the record. Unique key combination of Item, Warehouse, Calendar, Period, Cost Method, Component Class, Analysis Code and Cost Level will be ignored.	GMF_API_IGNORE_IC_UNIQUE_KEY
Warning: Item No &ITEM_NO is ignored since Item Id is supplied.	GMF_API_IGNORE_ITEM_NO
Warning: Lower level details cannot be updated or deleted hence ignored.	GMF_API_IGNORE_LOWER_LEVEL
Error: Invalid Accounting Unit in the Basis Account Key &BAS_ACC_KEY.	GMF_API_INVALID_ACCTG_UNIT
Error: Invalid Accounting No in the Basis Account Key &BAS_ACC_KEY.	GMF_API_INVALID_ACCT_NO
Error: Invalid combination of Allocation Code &ALLOC_CODE and Company &CO_CODE.	GMF_API_INVALID_ALLOC_CODE
Error: Allocation Method &ALLOC_METHOD not consistent with the existing method.	GMF_API_INVALID_ALLOC_DEF
Error: Invalid Allocation Id: &ALLOC_ID.	GMF_API_INVALID_ALLOC_ID

Message Name	Message Code
Error: Invalid Allocation Method: &ALLOC_METHOD.	GMF_API_INVALID_ALLOC_MTHD
Error: Invalid Analysis Code: &ANALYSIS_CODE.	GMF_API_INVALID_ANALYSIS_CODE
Error: Invalid Balance Type: &BALANCE_TYPE.	GMF_API_INVALID_BALANCE_TYPE
Error: Invalid Basis amount Indicator: &BAS_YTD_PTD.	GMF_API_INVALID_BAS_YTD_PTD
Error: Invalid Burden Indicator: &BURDEN_IND.	GMF_API_INVALID_BURDEN_IND
Error: Invalid Burden Quantity: &BURDEN_QTY (Should be between 0.00000001 and 999999999D99999999).	GMF_API_INVALID_BURDEN_QTY
Error: Invalid Burden UOM: &BURDEN_UM.	GMF_API_INVALID_BURDEN_UM
Error: Invalid Burden Usage: &BURDEN_USAGE (Should be between 0.00000001 and 999999999D99999999).	GMF_API_INVALID_BURDEN_USAGE
Error: Invalid Calendar Code: &CALENDAR_CODE.	GMF_API_INVALID_CALENDAR_CODE
Error: Invalid Component Class Code: &CMPNTCLS_CODE.	GMF_API_INVALID_CMPNTCLS_CODE
Error: Invalid Component Class Id: &CMPNTCLS_ID.	GMF_API_INVALID_CMPNTCLS_ID
Error: Component cost &CMPNT_COST should not be greater than 999,999,999.999999999.	GMF_API_INVALID_CMPNT_COST
Error: Invalid Cost Method Code: &COST_MTHD_CODE.	GMF_API_INVALID_COST_MTHD_CODE
Error: Invalid Delete Mark: &DELETE_MARK.	GMF_API_INVALID_DELETE_MARK
Error: Invalid Percentage: &FIXED_PERCENT (Should be 0 to 100).	GMF_API_INVALID_FIXED_PERCENT
Error: Invalid Item Id: &ITEM_ID.	GMF_API_INVALID_ITEM_ID
Error: Invalid Item No: &ITEM_NO.	GMF_API_INVALID_ITEM_NO
Error: Invalid Item Quantity: &ITEM_QTY (Should be between 0.00000001 and 999999999D99999999).	GMF_API_INVALID_ITEM_QTY

Message Name	Message Code
Error: Invalid Item UOM: &ITEM_UM.	GMF_API_INVALID_ITEM_UM
Error: Invalid Nominal Cost: &NOMINAL_COST (Should be between 0 and 999999999D999999999).	GMF_API_INVALID_NOMINAL_COST
Error: Invalid Organization Code: &ORGN_CODE.	GMF_API_INVALID_ORGN_CODE
Error: Invalid Period Code: &PERIOD_CODE for Calendar &CALENDAR_CODE.	GMF_API_INVALID_PERIOD_CODE
Error: Invalid Resource Code: &RESOURCES.	GMF_API_INVALID_RESOURCES
Error: For Actual Cost method, Raw Material Cost Calculation Type should be 1-PMAC, 2-PWAC, 3-PPAC, 4-LSTT or 5 -LSTI.	GMF_API_INVALID_RMCALC_TYPE
Error: Invalid Usage Unit of Measure: &USAGE_UM.	GMF_API_INVALID_USAGE_UM
Error: Invalid User name: &USER_NAME.	GMF_API_INVALID_USER_NAME
Error: Invalid Warehouse Code: &WHSE_CODE.	GMF_API_INVALID_WHSE_CODE
Error: Either Item Id or Item No is required.	GMF_API_ITEM_ID_REQ
Error: Item Quantity is required.	GMF_API_ITEM_QTY_REQ
Error: UOM conversion error for Item &ITEM_ID. Cannot convert quantity from current Item UOM &ITEM_UM to actual Item UOM &ITEM_ACT_UM.	GMF_API_ITEM_UM_CONV_ERR
Error: Item UOM is required.	GMF_API_ITEM_UM_REQ
Error: Line Number is required.	GMF_API_LINE_NO_REQ
Error: Nominal Cost is required.	GMF_API_NOMINAL_COST_REQ
Message: No rows deleted.	GMF_API_NO_ROWS_DEL
Error: No rows found to update/delete for Allocation Id &ALLOC_ID and Line No &LINE_NO.	GMF_API_NO_ROWS_FOUND
Message: No rows inserted.	GMF_API_NO_ROWS_INS
Message: No rows updated.	GMF_API_NO_ROWS_UPD
Error: Organization Code is required.	GMF_API_ORGN_CODE_REQ
Error: Period Code is required.	GMF_API_PERIOD_CODE_REQ

Message Name	Message Code
Error: Resource Code is required.	GMF_API_RESOURCES_REQ
&NUM_ROWS row(s) deleted.	GMF_API_ROWS_DEL
&NUM_ROWS row(s) inserted.	GMF_API_ROWS_INS
&NUM_ROWS row(s) updated.	GMF_API_ROWS_UPD
Error: No rows found to update/delete for Resource &RESOURCES, Orgn &ORGN_CODE, Calendar &CALENDAR_CODE, Period &PERIOD_CODE and Cost Method &COST_MTHD_CODE.	GMF_API_RSRC_NO_ROWS_FOUND
Warning: Total Fixed Percentage for Allocation Id &ALLOC_ID is not equal to 100.	GMF_API_TOTAL_PCT_NOTHUNDRED
Error: Burden UOM &BURDEN_UM must be of the same type as the resource UOM &RESOURCE_UM.	GMF_API_UOM_SAMETYPE_REQ
Error: No Component Cost Details found to update for Component Class Id &CMPNTCLS_ID and Analysis Code &ALYS_CODE.	GMF_API_UPD_IC_NOT_FOUND_DTL
Error: No Component Cost Details found to update for Component Cost Id &CMPNTCOST_ID.	GMF_API_UPD_IC_NOT_FOUND_ID
Error: Usage Unit of Measure is required.	GMF_API_USAGE_UM_REQ
Error: Warehouse Code is required.	GMF_API_WHSE_CODE_REQ

Glossary

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 APIs. 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, 4-16
 Delete, 1-5
 Insert, 1-5
 Structure, 3-4
 Structure and Logic, 4-17
 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, 4-22
 Header Entity, 4-21
 Insert, 1-5
 Retrieve, 1-6
 Structure, 3-5
 Structure and Logic, 4-23
 Update, 1-5
Burden Details APIs, 1-5
Business Object, 1
Business Objects, 4-1
 Allocation Definition, 4-16
 Burden Details, 4-21
 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_API.G_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
GMFPALCB.pls, 3-4
GMFPALCS.pls, 1-7, 3-4
GMFPBRDB.pls, 1-7, 3-5
GMFPBRDS.pls, 1-7, 3-5
GMFPCSTB.pls, 1-7, 3-1
GMFPCSTS.pls, 1-7, 3-1
GMFPRESB.pls, 3-3
GMFPRESS.pls, 1-7, 3-3
Group API, 2

I

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

M

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

P

p_allocation_definition_rec, 4-19
p_api_version, 3-6
p_commit, 3-7
p_dtl_tbl, 4-27, 4-28
p_header_rec, 4-9, 4-10, 4-11, 4-27, 4-28, 4-29
p_init_msg_list, 3-6
p_lower_level_dtl_tbl, 4-9, 4-10
p_resource_cost_rec, 4-15, 4-20
p_this_level_dtl_tbl, 4-9, 4-10
Package Specification - GMFPALCS.pls, 4-18
Package Specification - GMFPBRDS.pls, 4-24
Package Specification - GMFPCSTS.pls, 4-5
Package Specification - GMFPRESS.pls, 4-13
PL/SQL, 1-2
Private API, 1
processing standard message functions, 3-7
Public API, 1
Public Procedure Delete_Allocation_
 Definition, 4-20
Public Procedure Delete_Item_Cost, 4-10, 4-28

Public Procedure Delete_Resource_Cost, 4-15
Public Procedure Get_Burden_Details, 4-29
Public Procedure Get_Item_Cost, 4-11
Public Procedure Get_Resource_Cost, 4-15
Public Procedure Update_Allocation_
 Definition, 4-19
Public Procedure Update_Burden_Details, 4-27
Public Procedure Update_Item_Cost, 4-9
Public Procedure Update_Resource_Cost, 4-15

R

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

S

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

T

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

V

version compatibility, validation, 3-6

X

x_dtl_tbl, 4-29
x_lower_level_dtl_tbl, 4-11
x_msg_count, 3-7
x_msg_data, 3-7

x_resource_cost_rec, 4-15
x_return_status, 3-7
x_this_level_dtl_tbl, 4-11

