

# Oracle® Process Manufacturing

Product Development Recipe API User's Guide

Release 11i

**Part No. A97387-02**

January 2003

Oracle Process Manufacturing Product Development Recipe API User's Guide, Release 11i

Part No. A97387-02

Copyright © 2003 Oracle Corporation. All rights reserved.

Primary Author: Michele-Andrea Fields

Contributors: Shyam Sitaraman, Bill Stearns

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.

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 of Oracle Corporation. Other names may be trademarks of their respective owners.

---

---

# Contents

<b>Send Us Your Comments .....</b>	<b>vii</b>
<b>Preface.....</b>	<b>ix</b>
<b>Audience for This Guide.....</b>	<b>ix</b>
<b>How To Use This Guide .....</b>	<b>ix</b>
Documentation Accessibility .....	x
<b>Other Information Sources .....</b>	<b>x</b>
Online Documentation.....	x
Related User's Guides.....	xi
Guides Related to All Products .....	xi
User Guides Related to This Product .....	xi
Installation and System Administration .....	xvi
Other Implementation Documentation.....	xviii
Training and Support.....	xix
<b>Do Not Use Database Tools to Modify Oracle Applications Data.....</b>	<b>xx</b>
<b>About Oracle.....</b>	<b>xxi</b>
<b>Your Feedback .....</b>	<b>xxi</b>
<b>1 Recipe API Introduction</b>	
<b>Introducing the Recipe APIs .....</b>	<b>1-2</b>
Recipe API Support Policy .....	1-3
Technical Requirements .....	1-3
Technical Overview.....	1-3
Input Data Sources .....	1-4

Wrapper Function.....	1-6
Stored Procedure.....	1-6
Which API Functions are Included .....	1-7
Recipe API Bill of Materials .....	1-8

## 2 Recipe API Usage

<b>Creating a New Recipe</b> .....	2-1
<b>Using the Routing APIs</b> .....	2-4
<b>Using the Operation APIs</b> .....	2-5
<b>Using Input Parameters for Recipe APIs</b> .....	2-5

## 3 Technical Overview

<b>Recipe Header</b> .....	3-2
Structure for Recipe Header Public APIs .....	3-2
<b>Recipe Detail</b> .....	3-3
Structure for Recipe Detail Public APIs.....	3-3
<b>Recipe Validation</b> .....	3-5
Structure for Recipe Validation APIs.....	3-5
<b>Recipe Fetch</b> .....	3-6
Structure for Recipe Fetch APIs.....	3-6
<b>Recipe Validity Rule Fetch API</b> .....	3-7
Structure for Recipe Validity Rule Fetch APIs .....	3-7
<b>Routing API</b> .....	3-8
Structure for Routing APIs.....	3-8
<b>Routing Steps API</b> .....	3-9
Structure for Routing Steps APIs.....	3-9
<b>Routing Step Dependency API</b> .....	3-10
Structure for Routing Step Dependency APIs.....	3-10
<b>Operation API</b> .....	3-11
Structure for Operation APIs .....	3-11
<b>Operation Activity API</b> .....	3-12
Structure for Operation Activity APIs.....	3-12
<b>Operation Resources API</b> .....	3-13
Structure for Operation Resources APIs .....	3-13
<b>Activity API</b> .....	3-14

Structure for Activity APIs.....	3-14
<b>Standard Parameters</b> .....	3-15
Value-ID Conversion .....	3-16

## 4 Business Objects

<b>Recipe Header</b> .....	4-2
<b>Recipe Details</b> .....	4-9
<b>Recipe Validation</b> .....	4-20
<b>Recipe Fetch</b> .....	4-21
<b>Recipe Validity Rule Fetch</b> .....	4-22
<b>Routing</b> .....	4-24
<b>Routing Steps</b> .....	4-29
<b>Routing Step Dependency</b> .....	4-33
<b>Operation</b> .....	4-35
<b>Operation Activity</b> .....	4-39
<b>Operation Resources</b> .....	4-43
<b>Activity</b> .....	4-47

## A Messages and Errors

<b>Handling Messages</b> .....	A-2
<b>Interpreting Error Conditions</b> .....	A-5
<b>Understanding Error Messages</b> .....	A-5

## B How to Get Your OPM Recipe APIs Running

### Index



---

---

# Send Us Your Comments

**Oracle Process Manufacturing Product Development Recipe API User's Guide, Release 11i**  
**Part No. A97387-02**

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](mailto: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.



## Audience for This Guide

Welcome to Release 11i of the *Oracle Process Manufacturing Product Development Recipe API User's Guide*.

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

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

If you have never used *Oracle Process Manufacturing Product Development Recipe APIs*, Oracle suggests you attend one or more of the *Oracle Process Manufacturing Product Development Recipe APIs* training classes available through Oracle University.

- The Oracle Applications graphical user interface.

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

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

## How To Use This Guide

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

- Chapter 1 describes the Application Program Interfaces (APIs) that support external interfaces to the OPM Recipe tables.
- Chapter 2 describes the details of how to use the OPM Recipe APIs.
- Chapter 3 describes the structure of each OPM Recipe API.
- Chapter 4 provides the relationships between the OPM Recipe API table structure and its entities. It discusses Recipe 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.

- Appendix B provides a useful guide and examples for using the APIs.

## Documentation Accessibility

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

### Accessibility of Code Examples in Documentation

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

## Other Information Sources

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

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

## Online Documentation

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

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

- **11i Features Matrix** - This document lists new features available by patch and identifies any associated new documentation. The new features matrix document is available on MetaLink.
- **Readme File** - Refer to the readme file for patches that you have installed to learn about new documentation or documentation patches that you can download.

## Related User's Guides

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

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

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

## Guides Related to All Products

### Oracle Applications User's Guide

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

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

## User Guides Related to This Product

### Accounting Setup User's Guide

The OPM Accounting Setup application is where users set up global accounting attributes about the way financial data will be collected by OPM. These attributes include such things as account keys, financial calendars, and account segments.

Since OPM is closely integrated with Oracle General Ledger (GL), much of the attributes are defined in the Oracle GL instead of OPM, and therefore, the windows are display only within OPM. The *Oracle Process Manufacturing Accounting Setup User's Guide* describes how to setup and use this application.

### **Cost Management User's Guide**

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

### **Manufacturing Accounting Controller User's Guide**

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

### **Oracle Financials Integration User's Guide**

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

### **Inventory Management User's Guide**

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

## **Physical Inventory User's Guide**

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

## **Order Fulfillment User's Guide**

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

## **Purchase Management User's Guide**

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

## **Using Oracle Order Management with Process Inventory Guide**

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

## **Process Execution User's Guide**

The OPM Process Execution application lets you track firm planned orders and production batches from incoming materials through finished goods. Seamlessly integrated to the Product Development application, Process Execution lets you convert firm planned orders to single or multiple production batches, allocate

ingredients, record actual ingredient usage, and then complete and close production batches. Production inquiries and preformatted reports help you optimize inventory costs while maintaining a high level of customer satisfaction with on-time delivery of high quality products. The *OPM Process Execution User's Guide* presents overviews of the tasks and responsibilities for the Production Supervisor and the Production Operator. It provides prerequisite setup in other applications, and details the windows, features, and functionality of the OPM Process Execution application.

### **Integration with Advanced Planning and Scheduling User's Guide**

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

### **MPS/MRP and Forecasting User's Guide**

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

### **Capacity Planning User's Guide**

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

### **Using Oracle Process Manufacturing with Oracle Manufacturing Scheduling**

Oracle Process Manufacturing integrates with Oracle Manufacturing Scheduling to manage and utilize resources and materials. Through the Process Manufacturing application, you set up manufacturing, inventory, procurement and sales order data. Through the Manufacturing Scheduling application, you can optimize the

schedule based on resource and component constraints and user predefined priorities. Using different optimization objectives, you can tailor Manufacturing Scheduling to meet your needs.

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

### **Product Development User's Guide**

The Oracle Process Manufacturing Product Development application provides features to manage formula and laboratory work within the process manufacturing operation. It lets you manage multiple laboratory organizations and support varying product lines throughout the organization. You can characterize and simulate the technical properties of ingredients and their effects on formulas. You can optimize formulations before beginning expensive laboratory test batches. Product Development coordinates each development function and enables a rapid, enterprise-wide implementation of new products in your plants. The *Oracle Process Manufacturing Product Development User's Guide* describes how to setup and use this application.

### **Quality Management User's Guide**

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

### **Regulatory Management User's Guide**

The Oracle Process Manufacturing Regulatory Management application generates the Material Safety Data Sheets (MSDSs) required by authorities to accompany hazardous materials during shipping. You can create MSDSs from OPM Formula

Management with Regulatory or Production effectivities. The *Oracle Process Manufacturing Regulatory Management User's Guide* describes how to setup and use this application.

## **Implementation Guide**

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

## **System Administration User's Guide**

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

## **API User's Guides**

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

# **Installation and System Administration**

## **Oracle Applications Concepts**

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

also introduces the concepts behind Applications-wide features such as Business Intelligence (BIS), languages and character sets, and Self-Service Web Applications.

## **Installing Oracle Applications**

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

## **Upgrading Oracle Applications**

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

## **Maintaining Oracle Applications**

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

## **Oracle Applications System Administrator's Guide**

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

## **Oracle Alert User's Guide**

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

## **Oracle Applications Developer's Guide**

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards for Forms-Based Products*. It also provides information to help you build your custom Oracle Forms Developer 6i forms so that they integrate with Oracle Applications.

## **Oracle Applications User Interface Standards for Forms-Based Products**

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

## **Other Implementation Documentation**

### **Oracle Applications Product Update Notes**

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

### **Multiple Reporting Currencies in Oracle Applications**

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

### **Multiple Organizations in Oracle Applications**

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

### **Oracle Workflow Guide**

This guide explains how to define new workflow business processes as well as customize existing Oracle Applications-embedded workflow processes. You also use

this guide to complete the setup steps necessary for any Oracle Applications product that includes workflow-enabled processes.

### **Oracle Applications Flexfields Guide**

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

### **Oracle eTechnical Reference Manuals**

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

### **Oracle Manufacturing APIs and Open Interfaces Manual**

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

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

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

### **Oracle Applications Message Reference Manual**

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

## **Training and Support**

### **Training**

Oracle offers a complete set of training courses to help you and your staff master *Oracle Process Manufacturing Product Development Recipe APIs* and reach full

productivity quickly. These courses are organized into functional learning paths, so you take only those courses appropriate to your job or area of responsibility.

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

### Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep *Oracle Process Manufacturing Product Development Recipe APIs* working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle8i server, and your hardware and software environment.

## Do Not Use Database Tools to Modify Oracle Applications Data

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

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

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

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to

track who has changed your information because SQL\*Plus and other database tools do not keep a record of changes.

## About Oracle

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

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

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

## Your Feedback

Thank you for using *Oracle Process Manufacturing Product Development Recipe APIs* and this user's guide.

Oracle values your comments and feedback. At the end of this guide is a Reader's Comment Form you can use to explain what you like or dislike about *Oracle Process Manufacturing Product Development Recipe APIs* or this user's guide. Mail your comments to the following address or call us directly at (650) 506-7000.

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

Or, send electronic mail to **[appsdoc\\_us@oracle.com](mailto:appsdoc_us@oracle.com)**.



---

---

# Recipe API Introduction

This document describes the Application Program Interfaces (APIs) that support external interfaces to Oracle Process Manufacturing Product Development applications. The topics discussed in this chapter are:

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

## Introducing the Recipe APIs

Recipe APIs are business objects that can create, modify, or validate the recipe information. This document describes the usage of the business objects that are stored as PL/SQL packages within the OPM database schema, such as:

- Stored procedures used within these packages
- Parameters that these procedures accept and the values that return to the calling program
- Multi-Lingual support
- Error handling methodology

A Recipe is an entity that contains the minimum set of information that uniquely defines the manufacturing requirements for a specific product. Recipes provide a way to describe products and how these products are produced.

### What Is In This Document

This document describes the basic business needs, major features, architecture, and components for the Recipe APIs. Much of the application is divided into application-specific objects that lets you link OPM functionality into your own programs. The interfaces can make use of the standard functionality and logic implemented in the Product Development application.

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

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

## Recipe API Support Policy

Recipe APIs are supported by Oracle. This means:

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

## Technical Requirements

Recipe APIs are designed to operate in an OPM 11i environment only.

The procedure makes 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 for a 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 search the procedure messages.

These packages are part of the 11i Oracle Applications installation. Refer to the *Oracle Applications Coding Standards* guide for further details.

## Technical Overview

Recipe APIs are intended to be invoked from a user wrapper calling function with recipe related attributes passed to the procedure through a table or record format. The wrapper function is responsible for connecting to the database as an appropriate user with the necessary privileges. It passes the appropriate parameters into the stored procedure and is responsible for handling the return code from the procedure. OPM applications use windows as a wrapper function to call Recipe APIs.

Recipe APIs have the following procedures:

- Creation of new Recipe Header
- Creation of new Recipe Details, such as Recipe Customer details, Recipe Routing Step Details, Recipe Validity Rule details

- Fetch of Recipe and Validity Rules
- Modification of Recipe Header information
- Modification of Recipe Detail information
- Recipe validation package
- Creation of new activity
- Modification of activity
- Deletion of activity
- Creation of new operation (header and detail data)
- Modification of operation (header and detail data)
- Deletion of operation (header and detail data)

These stored procedures are called from a user wrapper function that executes the procedure and deals with the return status and messages from the execution.

## Input Data Sources

### Flat File

Input data to the user wrapper function comes from a flat file source. This is processed by the wrapper and header information, passed as parameters, to the stored procedure in a synchronous mode. However, along with the standard parameters, the header information is passed as a PL/SQL table. In this mode, the calling function monitors the success or failure (return code) from the called procedure. It also provides an option to COMMIT work done by the procedure.

### Batch File

Input data to the user wrapper function comes from a batch file. This is processed by the wrapper and header information passed, as parameters, to the stored procedure in an asynchronous mode. In this mode, the calling function does not monitor the success or failure of each individual record. The Oracle Message FND\_PUB\_MSG functionality is used to audit the calls.

### Online User Interface (UI)

Input data to the user wrapper function comes from a window or other user interface. This is processed by the UI and the details passed, as parameters, to the stored procedure in a synchronous mode. In this mode, the UI calling function

monitors the success or failure (return code) from the called procedure. It also provides an option to COMMIT work done by the procedure.

## Wrapper Function

Windows are generally used as wrapper functions.

The wrapper function is responsible for collating the details required as input parameters to the stored procedure, forwarding these in the call and monitoring the return code.

The stored procedure returns three possible return code:

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

Based on the return, the wrapper function searches the Oracle Messages File for the stored procedure to determine a COMMIT of the transaction or not.

## Stored Procedure

The stored procedure is called with the appropriate parameters forwarded in a PL/SQL table format. The procedure validates each record from this table and then processes the appropriate functional logic as required. The procedure writes appropriate messages to the Oracle Messages table. These are informational (success) or error as determined by the logic. These can be interrogated by the calling wrapper function through the GET MESSAGES functionality.

The stored procedure calls other validation procedures in the course of its execution; a modular approach has been adopted. Functions called by these procedures do not use IN/OUT parameters as these have been removed from the Oracle 8 coding standards.

On successful completion of the procedure, a success unit is in place that can be optionally COMMITTED. The decision as to whether a COMMIT is issued on successful completion is under the control of the calling code and deliberately outside the scope of the API procedures.

## Which API Functions are Included

The following are the Recipe API functions, the business object name, and a brief explanation of each function:

Function	Business Object Name	Description
Recipe Create	GMD_RECIPE_HEADER	Creates or modifies Recipe Header.
Recipe Detail	GMD_RECIPE_DETAIL	Creates or modifies the Recipe Detail information.
Recipe Validation	GMD_RECIPE_VAL	Validates the Recipe that gets created or validated.
Recipe Fetch	GMD_RECIPE_FETCH_PUB	Retrieves recipe details.
Recipe Validity Rule Fetch	GMD_FETCH_VALIDITY_RULES	Retrieves recipe validity rule details.
Routing	GMD_ROUTINGS_PUB	Creates, modifies, or deletes routing header and detail information. Can be used for modification of the routing header information.
Routing Steps	GMD_ROUTING_STEPS_PUB	Creates, modifies, or deletes the routing step information.
Routing Step Dependency	GMD_STEP_DEPENDENCY_PUB	Creates, modifies, or deletes the step dependency information.
Operation	GMD_OPERATION_PUB	Creates operation header and detail information. Can be used for modification and deletion of the operation header information.
Operation Activity	GMD_OPERATION_ACTIVITY_PUB	Creates, modifies, or deletes the operation detail information.
Operation Resources	GMD_OPERATION_RESOURCES_PUB	Creates, modifies, or deletes the operation resource information.
Activity	GMD_ACTIVITY_PUB	Creates, modifies, or deletes activity information.

## Recipe API Bill of Materials

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

Package Name	File Names	Description
GMD_RECIPE_HEADER	GMDPRCHS.pls, GMDPRCHB.pls	Public Recipe API that the user defined function calls.
GMD_RECIPE_DETAIL	GMDPRCDS.pls, GMDPRCDB.pls	Public Recipe API that the user defined function calls.
GMD_RECIPE_FETCH_PUB	GMDPRCFS.pls, GMDPRCFB.pls	Public Recipe API that Recipe Header and Recipe Detail APIs call.
GMD_FETCH_VALIDITY_RULES	GMDPVRFS.pls, GMDPVRFB.pls	Public Recipe API that Recipe Header and Recipe Detail APIs call.
GMD_RECIPE_COMMON_VAL	GMDRVALS.pls, GMDRVALB.pls	Public Recipe API that Recipe Header and Recipe Detail APIs call.
GMD_ROUTINGS_PUB	GMDPROUS.pls, GMDPROUB.pls	Public Routing package that the user defined function calls. The business API is used for creating, modifying, or deleting a routing header.
GMD_ROUTING_STEPS_PUB	GMDPRTSS.pls, GMDPRTSB.pls	Public Routing package that the user defined function calls. The business API is used for creating or modifying routing steps associated to the routing header.
GMD_STEP_DEPENDENCY_PUB	GMDPRTSS.pls, GMDPRTSB.pls	Public Routing package that the user defined function calls. The business API is used for creating or modifying routing step dependency information associated to the routing steps.

Package Name	File Names	Description
GMD_OPERATION_PUB	GMDPOPNS.pls, GMDPOPNB.pls	Public Operation package that the user defined function calls. The business API is used for creating, modifying, or deleting a operation header. When creating an operation header, the API also creates activities and resources associated with this header.
GMD_OPERATION_ACTIVITIES_PUB	GMDPOPAS.pls, GMDPOPAB.pls	Public Operation Activities package that the wrapper or user defined function calls. The business API is used for creating, modifying, or deleting operation activities. When creating an operation activity, the API also creates operation resources associated with this operation activity.
GMD_OPERATION_RESOURCES_PUB	GMDVOPRS.pls, GMDVOPRB.pls	Public Operation Resources package that the wrapper or user defined function calls. The business API is used for creating, modifying, or deleting operation resources.
GMD_ACTIVITY_PUB	GMDPACTB.pls, GMDPACTS.pls	Public Activity package that the user defined function calls. The business API is used for creating, modifying, or deleting activity information.



---

---

## Recipe API Usage

This topic describes how to use the Recipe APIs, including the steps to create a new recipe. The topics discussed in this chapter are:

- Create a New Recipe
- Using the Routing APIs
- Using the Operation APIs
- Using Input Parameters for Recipe APIs

### Creating a New Recipe

In order to use the recipe business object APIs effectively, follow these steps:

#### Step 1: Collate Recipe data into a PL/SQL table format

Recipe APIs are called through different types of sources. For example, you use an interface like Oracle Forms, or optionally create a wrapper function that calls these APIs. However, ensure that relevant recipe information is structured in a PL/SQL table format before passing it as a parameter to these APIs.

The APIs also require certain standard parameters such as API version and API name that need to be passed by the wrapper function. After performing the appropriate tasks, the APIs return the status code. Depending on this status code, the calling function decides to commit the work. If the return code is an error, then the function retrieves the error message text from the error stack.

#### Creating a PL/SQL table - Example

```
PROCEDURE on_insert IS
    X_hdr_tblgmd_recipe_header.recipe_tbl;
    X_hdr_flexgmd_recipe_header.recipe_flex;
```

```

X_statusVARCHAR2(30);
X_msg_cntNUMBER;
X_msg_datVARCHAR2(30);
X_rowNUMBER := 1;
BEGIN

X_hdr_tbl(X_row).recipe_id           := :gmd_recipes.recipe_id;
X_hdr_tbl(X_row).recipe_description  := :gmd_recipes.recipe_description;
X_hdr_tbl(X_row).recipe_no          := :gmd_recipes.recipe_no;
X_hdr_tbl(X_row).recipe_version     := :gmd_recipes.recipe_version;
X_hdr_tbl(X_row).user_id            := :gmd_recipes.created_by;
X_hdr_tbl(X_row).calculate_step_quantity
    := :gmd_recipes.calculate_step_quantity;
X_hdr_tbl(X_row).owner_orgn_code    := :gmd_recipes.owner_orgn_code;
X_hdr_tbl(X_row).creation_orgn_code := :gmd_recipes.creation_orgn_code;
X_hdr_tbl(X_row).formula_id         := :gmd_recipes.formula_id;
X_hdr_tbl(X_row).formula_no         := :gmd_recipes.formula_no;
X_hdr_tbl(X_row).formula_vers       := :gmd_recipes.formula_vers;
X_hdr_tbl(X_row).routing_id         := :gmd_recipes.routing_id;
X_hdr_tbl(X_row).routing_no         := :gmd_recipes.routing_no;
X_hdr_tbl(X_row).routing_vers       := :gmd_recipes.routing_vers;
X_hdr_tbl(X_row).project_id         := :gmd_recipes.project_id;
X_hdr_tbl(X_row).recipe_status      := :gmd_recipes.recipe_status;
X_hdr_tbl(X_row).planned_process_loss := :gmd_recipes.planned_process_loss;
X_hdr_tbl(X_row).owner_lab_type     := :gmd_recipes.owner_lab_type;
X_hdr_tbl(X_row).text_code          := :gmd_recipes.text_code;
X_hdr_tbl(X_row).delete_mark        := :gmd_recipes.delete_mark;
X_hdr_tbl(X_row).creation_date      := :gmd_recipes.creation_date;
X_hdr_tbl(X_row).created_by         := :gmd_recipes.created_by;
X_hdr_tbl(X_row).last_updated_by    := :gmd_recipes.last_updated_by;
X_hdr_tbl(X_row).last_update_date   := :gmd_recipes.last_update_date;
X_hdr_tbl(X_row).last_update_login  := :gmd_recipes.last_update_login;
X_hdr_tbl(X_row).owner_id           := :gmd_recipes.owner_id;
X_hdr_flex(X_row).attribute_category := :gmd_recipes.attribute_category;
X_hdr_flex(X_row).attribute1        := :gmd_recipes.attribute1;
X_hdr_flex(X_row).attribute2        := :gmd_recipes.attribute2;
X_hdr_flex(X_row).attribute3        := :gmd_recipes.attribute3;
X_hdr_flex(X_row).attribute4        := :gmd_recipes.attribute4;
X_hdr_flex(X_row).attribute5        := :gmd_recipes.attribute5;
X_hdr_flex(X_row).attribute6        := :gmd_recipes.attribute6;
X_hdr_flex(X_row).attribute7        := :gmd_recipes.attribute7;
X_hdr_flex(X_row).attribute8        := :gmd_recipes.attribute8;
X_hdr_flex(X_row).attribute9        := :gmd_recipes.attribute9;
X_hdr_flex(X_row).attribute10       := :gmd_recipes.attribute10;
X_hdr_flex(X_row).attribute11       := :gmd_recipes.attribute11;

```

```

X_hdr_flex(X_row).attribute12      := :gmd_recipes.attribute12;
X_hdr_flex(X_row).attribute13      := :gmd_recipes.attribute13;
X_hdr_flex(X_row).attribute14      := :gmd_recipes.attribute14;
X_hdr_flex(X_row).attribute15      := :gmd_recipes.attribute15;
X_hdr_flex(X_row).attribute16      := :gmd_recipes.attribute16;
X_hdr_flex(X_row).attribute17      := :gmd_recipes.attribute17;
X_hdr_flex(X_row).attribute18      := :gmd_recipes.attribute18;
X_hdr_flex(X_row).attribute19      := :gmd_recipes.attribute19;
X_hdr_flex(X_row).attribute20      := :gmd_recipes.attribute20;
X_hdr_flex(X_row).attribute21      := :gmd_recipes.attribute21;
X_hdr_flex(X_row).attribute22      := :gmd_recipes.attribute22;
X_hdr_flex(X_row).attribute23      := :gmd_recipes.attribute23;
X_hdr_flex(X_row).attribute24      := :gmd_recipes.attribute24;
X_hdr_flex(X_row).attribute25      := :gmd_recipes.attribute25;
X_hdr_flex(X_row).attribute26      := :gmd_recipes.attribute26;
X_hdr_flex(X_row).attribute27      := :gmd_recipes.attribute27;
X_hdr_flex(X_row).attribute28      := :gmd_recipes.attribute28;
X_hdr_flex(X_row).attribute29      := :gmd_recipes.attribute29;
X_hdr_flex(X_row).attribute30      := :gmd_recipes.attribute30;
gmd_recipe_header.create_recipe_header(p_api_version      => 1.0,
                                        p_init_msg_list     => 'F',
                                        p_commit            => 'F',
                                        p_called_from_forms => 'YES',
                                        x_return_status      => X_status,
                                        x_msg_count         => X_msg_cnt,
                                        x_msg_data          => X_msg_dat,
                                        p_recipe_header_tbl => X_hdr_tbl,
                                        p_recipe_header_flex => X_hdr_flex);

```

## Step 2: Call the Public API - Main validation performed

- a. The API checks for the existence of an appropriate User ID in the FND\_USR table. If there is no valid user, then the API puts error messages in an error stack, and prevents the creation of a recipe.
- b. The API checks for valid recipe number and version. For all updates or changes to the Recipe Header information, the recipe ID, or the recipe name and recipe version need to be provided. For changes to Recipe Detail, the recipe ID must be provided.
- c. After the Recipe Header is created successfully, Recipe Details are created.
- d. No changes or updates are performed for any recipes that have final cost updates done on them.

### **Step 3: Review any error messages**

The API returns the status code as one of its parameters after it is executed. The status code represents S for success, E for error, or U for Unexpected or Unknown status. If an error E occurs, then the calling function or wrapper function analyzes the errors from the message stack.

For more information on error messages, refer to "Appendix A".

### **Step 4: Check if relevant recipe information has been created**

After executing these APIs successfully, and committing the work, the recipe information needs to be tested using a different user interface. For example, after the recipe header and detail are created, commit the work, and run the appropriate Oracle application to test whether or not a new recipe is actually created.

## **Using the Routing APIs**

A routing is a sequenced set of operations, organized in steps, that must be performed to complete a production batch. A routing must include at least one routing step, or operation.

The Routing APIs are business objects that create or change the routing information. This information is comprised of:

- A header providing the routing number, version, description, status, routing class and description, valid from and to dates, quantity, unit of measure, planned loss percentage, owner, and owner organization.
- One or more detail lines for each header. Each detail line provides the routing step number, operation, operation version, operation description, step quantity, unit of measure, and release type.
  - Routing step dependency information can be set up to create and modify relationships between routing steps. This information includes previous step, dependency type, standard delay, and transfer percent data.
  - Prerequisite data: Operations, Activities, Resources

These actions are permissible on routings:

- Insert Routing (header and detail)
- Update Routing (header and detail)
- Delete Routing (header and detail)

## Using the Operation APIs

An operation is an ordered set of activities that must be completed for a step in a batch. An operation consists of at least one activity and its associated resources. An activity is a basic task performed at the plant.

The Operation APIs are business objects that create or change the operation or activity information. This information is comprised of:

- An activity record consisting of an activity code, description, and cost analysis code.
- A header providing the operation number, version, description, and status.
- One or more detail lines for each header. Each detail line provides the activity number, description, factor, offset interval, and other details for an operation.
  - Each activity can have associated resource information, including resource number, description, quantity, as well as throughput, cost, scheduling, process parameters.

These actions are permissible on activities:

- Insert Activity
- Update Activity
- Delete Activity

These actions are permissible on operations:

- Insert Operation (header and detail info)
- Update Operation (header and detail info)
- Delete Operation (header and detail info)

## Using Input Parameters for Recipe APIs

Most of the APIs requires data to be passed in a PL/SQL table or PL/SQL record type format. The wrapper functions, or Oracle Forms, create these tables or records by referencing the table or record types in the API specifications.

If the wrapper function needs to insert or update or delete Recipe Header detail, then it needs to create the following table types:

- RECIPE HEADER TABLE TYPE
- RECIPE INSERT FLEX OR RECIPE UPDATE FLEX

If the wrapper function needs to insert or update or delete Recipe Details, then it needs to create the following table types:

- RECIPE\_DETAIL\_TBL
- RECIPE\_MTL\_TBL
- RECIPE\_VR\_TBL

---

---

## Technical Overview

This topic describes the standards for the Recipe API structure. In addition, all APIs require standard parameters common to most APIs. This topic details the common parameters, as well as when they are used in an API. The topics discussed in this chapter are:

- „ Recipe Header
- „ Recipe Detail
- „ Recipe Validation
- „ Recipe Fetch
- „ Recipe Validity Rule Fetch
- „ Routing
- „ Routing Steps
- „ Routing Step Dependency
- „ Operation
- „ Operation Activity
- „ Operation Resources
- „ Activity
- „ Standard Parameters

## Recipe Header

Each function on the business object Recipe Header is associated with a Public API, through which Recipe Header details are created, updated, deleted, and retrieved from OPM.

The Public API performs all validations necessary on input data supplied to prevent the flow of invalid data into OPM.

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

### Structure for Recipe Header Public APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPRCHS.pls
Package Body File	GMDPRCHB.pls
Package	GMD_RECIPE_HEADER
Procedure - Create Recipe Header	Create_Recipe_Header
Procedure - Update Recipe Header	Update_Recipe_Header
Procedure - Delete Recipe Header	Delete_Recipe_Header

## Recipe Detail

Recipe Detail is associated with a Public API through which recipe details are created, updated, deleted, and retrieved from OPM.

The Public API performs all validations necessary on input data supplied to prevent the flow of invalid data into OPM.

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

### Structure for Recipe Detail Public APIs

Object Type	Name
Package Specification File	GMDPRCDS.pls
Package Body File	GMDPRCDB.pls
Package	GMD_RECIPE_DETAILS
Procedure - CREATE_RECIPE_PROCESS_LOSS	CREATE_RECIPE_PROCESS_LOSS
Procedure - CREATE_RECIPE_CUSTOMERS	CREATE_RECIPE_CUSTOMERS
Procedure - CREATE_RECIPE_VR	CREATE_RECIPE_VR
Procedure - UPDATE_RECIPE_VR	UPDATE_RECIPE_VR
Procedure - CREATE_RECIPE_MTL	CREATE_RECIPE_MTL
Procedure - UPDATE_RECIPE_PROCESS_LOSS	UPDATE_RECIPE_PROCESS_LOSS
Procedure - UPDATE_RECIPE_CUSTOMERS	UPDATE_RECIPE_CUSTOMERS
Procedure - RECIPE_ROUTING_STEPS	RECIPE_ROUTING_STEPS
Procedure - RECIPE_ORGN_OPERATIONS	RECIPE_ORGN_OPERATIONS
Procedure - RECIPE_ORGN_RESOURCES	RECIPE_ORGN_RESOURCES

The last three procedures, `RECIPE_ROUTING_STEPS`, `RECIPE_ORGN_OPERATIONS`, and `RECIPE_ORGN_RESOURCES` are used only if the routing information needs to be overridden at the recipe level.

## Recipe Validation

Recipe Validation API is used in conjunction with other public APIs like Recipe Header and Recipe Detail. The Public API performs all validations necessary to prevent the flow of invalid data into OPM.

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

### Structure for Recipe Validation APIs

Object Type	Name
Package Specification File	GMDRVALS.pls
Package Body File	GMDRVALB.pls
Package	GMD_RECIPE_VAL
Procedure - RECIPE_CUST_EXISTS	RECIPE_CUST_EXISTS
Procedure - RECIPE_DESCRIPTION	RECIPE_DESCRIPTION
Procedure - PROCESS_LOSS_FOR_UPDATE	PROCESS_LOSS_FOR_UPDATE
Procedure - RECIPE_EXISTS	RECIPE_EXISTS
Procedure - RECIPE_FOR_UPDATE	RECIPE_FOR_UPDATE
Procedure - RECIPE_NAME	RECIPE_NAME
Procedure - RECIPE_ORGN_CODE	RECIPE_ORGN_CODE

## Recipe Fetch

Recipe Fetch API retrieves recipe details. This Public API is called by other OPM applications like Process Planning, Process Execution, and Costing.

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

### Structure for Recipe Fetch APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPRCFS.pls
Package Body File	GMDPRCFB.pls
Package	GMD_RECIPE_FETCH_PUB
Procedure - GET_FORMULA_ID	GET_FORMULA_ID
Procedure - GET_OPRN_RESC_DETL	GET_OPRN_ACT_DETL
Procedure - GET_OPRN_RESC_DETL	GET_OPRN_RESC_DETL
Procedure - GET_PROCESS_LOSS	GET_PROCESS_LOSS
Procedure - GET_RECIPE_ID	GET_RECIPE_ID
Procedure - GET_RECIPE_STEP_DETAILS	GET_RECIPE_STEP_DETAILS
Procedure - GET_ROUTING_ID	GET_ROUTING_ID
Procedure - GET_ROUT_HDR	GET_ROUT_HDR
Procedure - GET_ROUT_MATERIAL	GET_ROUT_MATERIAL
Procedure - GET_STEP_DEPD_DETAILS	GET_STEP_DEPD_DETAILS

## Recipe Validity Rule Fetch API

Recipe Validity Rule Fetch API retrieves recipe validity rule details. This Public API is called by other OPM applications like Process Planning, Process Execution, and Costing.

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

### Structure for Recipe Validity Rule Fetch APIs

Object Type	Name
Package Specification File	GMDPVRF5.pls
Package Body File	GMDPVRF6.pls
Package	GMD_FETCH_VALIDITY_RULES
Procedure - GET_BATCHFORMULA_RATIO	GET_BATCHFORMULA_RATIO
Procedure - GET_CONTRIBUTING_QTY	GET_CONTRIBUTING_QTY
Procedure - GET_INPUT_RATIO	GET_INPUT_RATIO
Procedure - GET_INGREDPROD_RATIO	GET_INGREDPROD_RATIO
Procedure - GET_OUTPUT_RATIO	GET_OUTPUT_RATIO
Procedure - GET_VALIDITY_RULES	GET_VALIDITY_RULES
Procedure - UOM_CONVERSION_MESG	UOM_CONVERSION_MESG

## Routing API

The Routing API creates, modifies, or deletes the routing header and detail information. It can be used for modification of the routing header information.

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

### Structure for Routing APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPROUS.pls
Package Body File	GMDPROUB.pls
Package	GMD_ROUTING_PUB
Procedure - Create Routing	INSERT_ROUTING
Procedure - Modify Routing	UPDATE_ROUTING
Procedure - Delete Routing	DELETE_ROUTING
Procedure - Undelete Routing	UNDELETE_ROUTING

## Routing Steps API

The Routing Steps API creates, modifies, or deletes the routing step information.

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

### Structure for Routing Steps APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPRTSS.pls
Package Body File	GMDPRTSB.pls
Package	GMD_ROUTING_STEPS_PUB
Procedure - Create Routing Steps	INSERT_ROUTING_STEPS
Procedure - Modify Routing Steps	UPDATE_ROUTING_STEPS
Procedure - Delete Routing Steps	DELETE_ROUTING_STEP

## Routing Step Dependency API

The Routing Step Dependency API creates, modifies, or deletes the step dependency information.

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

### Structure for Routing Step Dependency APIs

Object Type	Name
Package Specification File	GMDPRTSS.pls
Package Body File	GMDPRTSB.pls
Package	GMD_ROUTING_STEPS_PUB
Procedure - Create Routing Step Dependency	INSERT_STEP_DEPENDENCIES
Procedure - Modify Routing Step Dependency	UPDATE_STEP_DEPENDENCIES
Procedure - Delete Routing Step Dependency	DELETE_STEP_DEPENDENCIES

## Operation API

The Operation API creates operation header and detail information. It can be used for modification and deletion of the operation header information.

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

### Structure for Operation APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPOPSS.pls
Package Body File	GMDPOPSB.pls
Package	GMD_OPERATIONS_PUB
Procedure - Create Operation	INSERT_OPERATION
Procedure - Modify Operation	UPDATE_OPERATION
Procedure - Delete Operation	DELETE_OPERATION

## Operation Activity API

The Operation Activity API creates, modifies, or deletes the operation activity information.

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

### Structure for Operation Activity APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPOPaS.pls
Package Body File	GMDPOPAB.pls
Package	GMD_OPERATION_ACTIVITIES_PUB
Procedure - Create Operation Activity	INSERT_OPERATION_ACTIVITY
Procedure - Modify Operation Activity	UPDATE_OPERATION_ACTIVITY
Procedure - Delete Operation Activity	DELETE_OPERATION_ACTIVITY

## Operation Resources API

The Operation Resources API creates, modifies, or deletes operation resources information.

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

### Structure for Operation Resources APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDPOPRS.pls
Package Body File	GMDPOPRB.pls
Package	GMD_OPERATION_RESOURCES_PUB
Procedure - Create Operation Resources	INSERT_OPERATION_RESOURCES
Procedure - Modify Operation Resources	UPDATE_OPERATION_RESOURCES
Procedure - Delete Operation Resources	DELETE_OPERATION_RESOURCES

## Activity API

The Activity API creates, modifies, or deletes activity information.

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

### Structure for Activity APIs

<b>Object Type</b>	<b>Name</b>
Package Specification File	GMDFACTB.pls
Package Body File	GMDFACTS.pls
Package	GMD_ACTIVITIES_PUB
Procedure - Create Activity	INSERT_ACTIVITY
Procedure - Modify Activity	UPDATE_ACTIVITY
Procedure - Delete Activity	DELETE_ACTIVITY

## Standard Parameters

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

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

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

Standard parameters are included in all APIs whenever applicable.

Standard IN parameters:

- `p_api_version`
- `p_init_msg_list`
- `p_commit`
- `p_validation_level`

Standard OUT parameters:

- `x_return_status`
- `x_msg_count`
- `x_msg_data`

Parameter	Type	IN/OUT	Required	Validation
<code>p_api_version</code>	varchar2	IN	Y	Validates version compatibility. The version sent by the calling function is compared to the internal version of the API and an unexpected error (U) is generated if these do not match.
<code>p_init_msg_list</code>	varchar2	IN	N	Used to specify whether the message list is initialized on entry to the API. It is an optional parameter, and supplied defaults to <code>FND_API.G_FALSE</code> that means that the API does not initialize the message list.

Parameter	Type	IN/OUT	Required	Validation
p_commit	varchar2	IN	N	Used to specify whether the API commits its work before returning to the calling function. If not supplied, then it defaults to FND_API.G_FALSE.
p_called_from_forms	varchar2	N	N	Set to Y (Yes) if the API is called from Oracle Forms. Set to N if called from any other third party function.
x_return_status	varchar2	OUT	-	Specifies whether the API was successful or failed. Valid values are S - Successful, E - failed due to expected error, U - failed due to unexpected error.
x_msg_count	number	OUT	-	Specifies the 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 the Business Object API Coding Standards Document.

## Value-ID Conversion

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

For example, an item is represented by an ID, 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, the `varchar2` column `item_no`. This value is displayed when you choose an item. Therefore, an item can be identified by either its ID or value, in this case `item_no`.

The following set of rules are for the conversion process:

- Either ID or value, or both can be passed to an API. But, when both values are passed, ID based parameters take precedence over value based parameters. For example, 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 informs the API caller that some of the input has been ignored.

- This message is not an error message. The API continues with its regular processing.
- Each value has to resolve into one ID. Failure to resolve a value into an ID is an error and is associated with an error message. The API aborts processing and returns with a return status of error.



---

---

## Business Objects

This topic describes the business object details for each Recipe API, including the associated procedures. Each API has both required and optional parameters, as well as available flexfields. The topics discussed in this chapter are:

- „ Recipe Header
- „ Recipe Details
- „ Recipe Validation
- „ Recipe Fetch
- „ Recipe Validity Rule Fetch
- „ Routing
- „ Routing Steps
- „ Routing Step Dependency
- „ Operation
- „ Operation Activity
- „ Operation Resources
- „ Activity

## Recipe Header

The Recipe Header associates a formula and routing to a product item. To create a recipe, it must have the formula information; however, routing details are optional. Based on the recipe header details, other recipe details like recipe customer, and recipe validity rules can be created.

When creating the recipe header, the table type `Recipe_header_tbl_type` is used as a parameter.

### Recipe Header Table Type Attributes

Field/column	Type	Length	Required	Description
RECIPE_ID	Number	15	N	If the recipe ID is passed, and a new recipe is created, then it needs to check if this recipe ID already exists in the database.
RECIPE_DESCRIPTION	Varchar2	40	Y	Recipe description.
RECIPE_NO	Varchar2	32	Y	Not required if recipe ID is passed.
RECIPE_VERSION	Number	15	Y	Not required if recipe ID is passed.
USER_ID	Number	15	N	ID for user derived from the <code>fnl_user</code> table. This is required if <code>user_name</code> is not provided.
USER_NAME	Varchar2	32	Y	User Name - usually the person who creates or modifies the recipe.
OWNER_ORGN_CODE	Varchar2	4	Y	Organization derived from table <code>sy_orgn_mst</code> for the user who owns this recipe.
CREATION_ORGN_CODE	Varchar2	4	Y	Organization derived from table <code>sy_orgn_mst</code> for the user who creates this recipe.
FORMULA_ID	Number	15	N	Formula ID.
FORMULA_NO	Varchar2	32	Y	Formula number associated with this recipe.

Field/column	Type	Length	Required	Description
FORMULA_VERS	Number	15	N	Formula version associated with this recipe.
ROUTING_ID	Number	15	N	Routing ID.
ROUTING_NO	Varchar2	32	N	Routing number associated with this recipe.
ROUTING_VERS	Number	15	Y	Routing version associated with this recipe.
PROJECT_ID	Number	15	N	Not currently used.
RECIPE_STATUS	Varchar2	16	Y	Status code for this Recipe. When a recipe is created, it defaults to new status code 100. The recipe status helps in the regulation of a workflow process to monitor the recipe development through different stages.
PLANNED_PROCESS_LOSS	Number	variable length	N	Process loss associated for the routing at recipe level.
TEXT_CODE	Number	variable length	N	Edit text code.
DELETE_MARK	Number	variable length	Y	This flag is used in logical deletion of Recipe Header.
CREATION_DATE	Date	variable length	Y	Standard Who Column.
CREATED_BY	Number	variable length	Y	Standard Who Column.
LAST_UPDATED_BY	Number	variable length	Y	Standard Who Column.
LAST_UPDATE_DATE	Date	variable length	N	Standard Who Column.
LAST_UPDATE_LOGIN	Number	variable length	Y	Standard Who Column.
OWNER_ID	Number	variable length	Y	User ID for the person who owns this recipe.
OWNER_LAB_TYPE	Varchar2	4	N	Lab organization.

Field/column	Type	Length	Required	Description
CALCULATE_STEP_QUANTITY	Number	variable length	N	Flag used for automatically calculating the step quantity value.

### Insert Flexfields Table Type Attributes

This table type specifies a list of flexfield attribute columns. By default, the values for these columns are set to NULL.

Field/column	Type	Length	Required	Description
ATTRIBUTE_CATEGORY	Varchar2	240	N	Descriptive Flexfield Segment Category.
ATTRIBUTE1	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE2	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE3	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE4	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE5	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE6	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE7	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE8	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE9	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE10	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE11	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE12	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE13	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE14	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE15	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE16	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE17	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE18	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE19	Varchar2	240	N	Descriptive Flexfield Segment.

Field/column	Type	Length	Required	Description
ATTRIBUTE20	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE21	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE22	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE23	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE24	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE25	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE26	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE27	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE28	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE29	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE30	Varchar2	240	N	Descriptive Flexfield Segment.

### Update Flexfields Table Type Attributes

This table type specifies a list of flexfield attribute columns. By default, the values for these columns are set to FND\_API.G\_MISS\_CHAR.

Field/column	Type	Length	Required	Description
ATTRIBUTE_CATEGORY	Varchar2	240	N	Descriptive Flexfield Segment Category.
ATTRIBUTE1	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE2	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE3	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE4	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE5	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE6	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE7	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE8	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE9	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE10	Varchar2	240	N	Descriptive Flexfield Segment.

<b>Field/column</b>	<b>Type</b>	<b>Length</b>	<b>Required</b>	<b>Description</b>
ATTRIBUTE11	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE12	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE13	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE14	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE15	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE16	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE17	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE18	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE19	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE20	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE21	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE22	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE23	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE24	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE25	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE26	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE27	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE28	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE29	Varchar2	240	N	Descriptive Flexfield Segment.
ATTRIBUTE30	Varchar2	240	N	Descriptive Flexfield Segment.

### Procedure CREATE\_RECIPE\_HEADER

Accepts recipe and flexfield information in a PL/SQL table format. The API creates the recipe header details. The API returns a success status (x\_return\_status = S) after creating the recipes.

Parameters	In/Out	Type	Required	Description
P_RECIPE_HEADER_TBL	IN	Recipe Header table type	Y	This table type comprises columns that capture all recipe header related information.
P_RECIPE_HEADER_FLEX	IN	Recipe Insert Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

### Procedure DELETE\_RECIPE\_HEADER

Accepts recipe and flexfield information in a PL/SQL table format. The API deletes the Recipe Header details. The API returns a success status (x\_return\_status = S) after it deletes the recipes. Since the delete on Recipe Header is a logical delete, it is equivalent to an update with the delete mark set to 1.

Parameters	In/Out	Type	Required	Description
P_RECIPE_HEADER_TBL	IN	Recipe Header table type	Y	This table type comprises columns that capture all recipe header related information.
P_RECIPE_UPDATE_FLEX	IN	Recipe Update Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

**Procedure UPDATE\_RECIPE\_HEADER**

Accepts recipe and flexfield information in a PL/SQL table format. The API updates the Recipe Header details. The API returns a success status (x\_return\_status = S) after it updates the recipes.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

Parameters	In/Out	Type	Required	Description
P_RECIPE_HEADER_TBL	IN	Recipe Header table type	Y	This table type comprises columns that capture all recipe header related information.
P_RECIPE_UPDATE_FLEX	IN	Recipe Update Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

## Recipe Details

Recipe Details defines such information as the customers, validity rules, and routing steps associated to the created recipe. To create Recipe Details you must have the Recipe Header details. In creating the recipe details, the table type `Recipe_detail_tbl_type` is used as a parameter.

### Recipe Details Table Type Attributes

Field/column	Type	Length	Required	Description
RECIPE_ID	Number	15	N	If the recipe ID is passed, and a new recipe is created, then it needs to check if this recipe ID already exists in the database.
RECIPE_NO	Varchar2	32	Y	Recipe name.
RECIPE_VERSION	Number	15	Y	Recipe version.
USER_ID	Number	15	N	ID for user derived from the <code>fnd_user</code> table. This is required if <code>user_name</code> is not provided.
USER_NAME	Varchar2	32	Y	User Name - usually the person who creates or modifies the recipe.
ORGN_CODE	Varchar2	4	N	Organization derived from table <code>sy_orgn_mst</code> for the user who owns this recipe. This is required if the recipe process loss details need to be created.
RECIPE_PROCESS_LOSS_ID	Number	15	N	Recipe process loss ID.
PROCESS_LOSS	Number	variable length	N	Process loss value for a Recipe Routing.
ACTIVITY_FACTOR	Number	variable length	N	Number of times the same activity is repeated in an operation.
MAX_CAPACITY	Number	variable length	N	Maximum Capacity for a given resource.

Field/column	Type	Length	Required	Description
MIN_CAPACITY	Number	variable length	N	Minimum Capacity for a given resource.
PROCESS_PARAMETER_1	Varchar2	16	N	Additional attributes for a resource. For example, fan rotation speed in rpm.
PROCESS_PARAMETER_2	Varchar2	16	N	Additional attributes for a resource. For example, fan rotation speed in rpm.
PROCESS_PARAMETER_3	Varchar2	16	N	Additional attributes for a resource. For example, fan rotation speed in rpm.
PROCESS_PARAMETER_4	Varchar2	16	N	Additional attributes for a resource. For example, fan rotation speed in rpm.
PROCESS_PARAMETER_5	Varchar2	16	N	Additional attributes for a resource. For example, fan rotation speed in rpm.
CUSTOMER_ID	Number	15	N	Customer ID.
CUSTOMER_NO	Varchar2	32	N	Customer Name associated with this recipe. This is required only when you want to associate a recipe to a customer.
ROUTINGSTEP_ID	Number	15	N	Routing Step ID.
OPRN_LINE_ID	Number	15	N	Operation line ID.
RESOURCES	Varchar2	16	N	Resource information.
PROCESS_UOM	Varchar2	4	N	Unit of measure for the Recipe operation.
USAGE_UM	Varchar2	4	N	Unit of measure for the resource usage.
RESOURCE_USAGE	Number	variable length	N	Resource Usage quantity.
PROCESS_QTY	Number	variable length	N	Operation quantity.

Field/column	Type	Length	Required	Description
STEP_QTY	Number	variable length	N	Routing Step quantity. This is required if you want to override the existing routing quantity at the recipe level.
MASS_QTY	Number	variable length	N	Routing Step quantity in mass unit of measure.
MASS_REF_UOM	Varchar2	4	N	Unit of measure for Mass type.
VOLUME_QTY	Number	variable length	N	Routing Step quantity in volume unit of measure.
VOLUME_REF_UOM	Varchar2	4	N	Unit of measure for volume type.
TEXT_CODE	Number	variable length	N	Edit text code.
DELETE_MARK	Number	variable length	Y	This flag is used in logical deletion of Recipe header.
CREATION_DATE	Date	variable length	Y	Standard Who Column.
CREATED_BY	Number	15	Y	Standard Who Column.
LAST_UPDATED_BY	Number	15	Y	Standard Who Column.
LAST_UPDATE_DATE	Date	variable length	N	Standard Who Column.
LAST_UPDATE_LOGIN	Number	15	Y	Standard Who Column.
ITEM_ID	Number	15	Y	Item ID.
OWNER_ID	Number	15	Y	User ID for the person who owns this recipe.

### Parameter Specification for Recipe Material Lines table type

This table type defines columns that associate the recipe material step with the corresponding formula line.

Field/column	Type	Length	Required	Description
RECIPE_ID	Number	15	Y	Recipe ID.
RECIPE_NO	Varchar2	32	Y	Recipe name.
RECIPE_VERSION	Number	15	Y	Recipe version.
USER_ID	Number	variable length	Y	User ID.
USER_NAME	Varchar2	variable length	Y	User creating the Recipe Material lines.
FORMULALINE_ID	Number	15	Y	Formula Line associated with the routing step.
TEXT_CODE	Number	variable length	N	Edit text code.
CREATION_DATE	Date	date	Y	Standard Who Column.
CREATED_BY	Number	variable length	Y	Standard Who Column.
LAST_UPDATED_BY	Number	variable length	Y	Standard Who Column.
LAST_UPDATE_DATE	Date	date	Y	Standard Who Column.
LAST_UPDATE_LOGIN	Number	variable length	Y	Standard Who Column.
ROUTINGSTEP_ID	Number	variable length	Y	Routing Step ID.

### Parameter Specification for Recipe Validity Rule table type

This table type defines the recipe validity rule details.

Field/column	Type	Length	Required	Description
RECIPE_VALIDITY_RULE_ID	Number	15	Y	Recipe validity rule ID.
RECIPE_ID	Number	15	Y	Recipe ID.
RECIPE_NO	Varchar2	32	Y	Recipe name.
RECIPE_VERSION	Number	15	Y	Recipe version.
USER_ID	Number	15	N	User ID. This is required if user_name is not provided.
USER_NAME	Varchar2	32	Y	User details.
ORGN_CODE	Varchar2	4	N	Organization code.
ITEM_ID	Number	15	Y	Item ID.
ITEM_NO	Varchar2	32	N	Item details. This is required if item_id is not provided.
RECIPE_USE	Varchar2	30	Y	Usage of recipe for planning, costing, and production.
PREFERENCE	Number	15	Y	Preference.
START_DATE	Date	variable length	Y	Valid effective start date.
END_DATE	Date	variable length	N	Valid effective end date.
MIN_QTY	Number	variable length	Y	Minimum quantity.
MAX_QTY	Number	variable length	Y	Maximum quantity.
STD_QTY	Number	variable length	Y	Standard quantity.
ITEM_UM	Varchar2	4	Y	Item unit of measure.
INV_MIN_QTY	Number	variable length	Y	Item quantity in its primary unit of measure.

Field/column	Type	Length	Required	Description
INV_MAX_QTY	Number	variable length	Y	Item quantity in its primary unit of measure.
TEXT_CODE	Number	variable length	N	Edit text code.
CREATED_BY	Number	15	Y	Standard Who Column.
CREATION_DATE	Date	variable length	Y	Standard Who Column.
LAST_UPDATED_BY	Number	15	Y	Standard Who Column.
LAST_UPDATE_DATE	Date	variable length	Y	Standard Who Column.
LAST_UPDATE_LOGIN	Number	15	N	Standard Who Column.
DELETE_MARK	Number	variable length	Y	Logical flag to delete a row in this table.
VALIDITY_RULE_STATUS	Varchar2	30	Y	Status code for a validity rule.

### Procedure CREATE\_RECIPE\_CUSTOMERS

Accepts recipe detail information in a PL/SQL table format. The API associates customers with the recipe details. The API returns a success status (x\_return\_status = S) after it creates the recipes customers information in the gmd\_recipe\_customers table.

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.

### Procedure UPDATE\_RECIPE\_CUSTOMERS

Accepts recipe detail information in a PL/SQL table format. The API updates customers with the recipe details. The API returns a success status (x\_return\_status = S) after it updates the recipes customers information in the gmd\_recipe\_customers table.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.

### Procedure CREATE\_RECIPE\_MTL

Associates the Recipe Routing step with the formula line information. The association details are passed in a PL/SQL table format. The API returns a success status (x\_return\_status = S) after creating the recipes step material details in the gmd\_recipe\_step\_materials table.

Parameters	In/Out	Type	Required	Description
P_RECIPE_MTL_TBL	IN	Recipe Material lines table type	Y	This table type comprises columns that capture all recipe step material and formula line related information.

### Procedure UPDATE\_RECIPE\_PROCESS\_LOSS

Updates the recipe specific planned process loss value in the gmd\_recipe\_process\_loss table.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.

**Procedure UPDATE\_RECIPE\_VR**

Updates the recipe validity rules.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

Parameters	In/Out	Type	Required	Description
P_RECIPE_VR_TBL	IN	Recipe Validity Rule table type	Y	This table type comprises columns that capture all recipe validity rule related information.
P_RECIPE_UPDATE_FLEX	IN	Recipe Update Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

**Procedure CREATE\_RECIPE\_PROCESS\_LOSS**

Accepts recipe process loss information in a PL/SQL table format. The process losses specific to organizations are saved in the gmd\_recipe\_process\_loss table using this API.

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.

**Procedure CREATE\_RECIPE\_VR**

Accepts recipe validity rule and flexfield information in a PL/SQL table format. The API return a success status after it create the validity rules in the GMD\_RECIPE\_VALIDITY\_RULES table.

Parameters	In/Out	Type	Required	Description
P_RECIPE_VR_TBL	IN	Recipe Validity Rule table type	Y	This table type comprises columns that capture all recipe validity rule related information.
P_RECIPE_INSERT_FLEX	IN	Recipe Insert Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

### Procedure **RECIPE\_ORGN\_OPERATIONS**

Accepts Recipe detail information in a PL/SQL table format. It also accepts either the insert or update flexfield PL/SQL table format. The update or insert operation depends on the values passed using these PL/SQL tables. For example, if the primary key values for the `gmd_recipe_orgn_operations` are passed, then the API performs an update. Otherwise, it creates a new row in the `gmd_recipe_orgn_operations` table. It returns a success status (`x_return_status = 'S'`) after performing an insert or update successfully.

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.
P_RECIPE_INSERT_FLEX	IN	Recipe Insert Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.
P_RECIPE_UPDATE_FLEX	IN	Recipe Update Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

### Procedure **RECIPE\_ORGN\_RESOURCES**

Accepts Recipe detail information in a PL/SQL table format. It also accepts either the insert or update flexfield PL/SQL table format. The update or insert resources depends on the values passed using these PL/SQL tables. For example, if the primary key values for the `gmd_recipe_orgn_operations` are passed, then the API performs an update. Otherwise, it creates a new row in the `gmd_recipe_orgn_resources` table. It returns a success status (`x_return_status = 'S'`) after performing an insert or update successfully.

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.
P_RECIPE_INSERT_FLEX	IN	Recipe insert flexfield table type	Y	This table type comprises columns that capture all flexfield related information.
P_RECIPE_UPDATE_FLEX	IN	Recipe Update flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

### Procedure **RECIPE\_ROUTING\_STEPS**

Accepts Recipe detail information in a PL/SQL table format. It also accepts either the insert or update flexfield PL/SQL table format. The update or insert recipe routing steps depends on the values passed using these PL/SQL tables. For example, if the primary key values for the `gmd_recipe_routing_steps` are passed, then the API performs an update. Otherwise, it creates a new row in the `gmd_recipe_routing_steps` table. It returns a success status (`x_return_status = 'S'`) after performing an insert or update successfully.

---

Parameters	In/Out	Type	Required	Description
P_RECIPE_DETAIL_TBL	IN	Recipe Detail table type	Y	This table type comprises columns that capture all recipe detail related information.
P_RECIPE_INSERT_FLEX	IN	Recipe Insert Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.
P_RECIPE_UPDATE_FLEX	IN	Recipe Update Flexfield table type	Y	This table type comprises columns that capture all flexfield related information.

## Recipe Validation

Common recipe validation package that other Recipe APIs call.

### **Procedure PROCESS\_LOSS\_FOR\_UPDATE**

Validation done prior to update on recipe process loss.

### **Procedure RECIPE\_CUST\_EXISTS**

This procedure checks if given ID or name and version exist in the gmd\_recipe\_customers table.

### **Procedure RECIPE\_DESCRIPTION**

This procedure checks if the recipe description is provided.

### **Procedure RECIPE\_EXISTS**

This procedure checks if given ID or name and version exist in gmd\_recipes\_b. If name and version is provided, then the ID is returned.

### **Procedure RECIPE\_FOR\_UPDATE**

Validation prior to update of a recipe.

### **Procedure RECIPE\_NAME**

This procedure checks if given name and version exist in gmd\_recipes\_b.

### **Procedure RECIPE\_ORGN\_CODE**

This procedure validates that a given orgn\_code is a plant or lab and that it is associated with the given user.

## Recipe Fetch

API to fetch recipe details. This API is called extensively by other applications, such as Process Execution, Process Planning, and Costing.

### **Procedure GET\_FORMULA\_ID**

This procedure, based on the recipe ID or recipe number and version, returns the formula ID that is associated with this recipe.

### **Procedure GET\_OPRN\_ACT\_DETL**

This procedure, based on recipe ID, returns the operation activities associated with this recipe.

### **Procedure GET\_OPRN\_RESC\_DETL**

This procedure, based on recipe ID, returns the operation resources associated with this recipe.

### **Procedure GET\_PROCESS\_LOSS**

This procedure, based on recipe ID or recipe number and version, returns the recipe process loss associated with this recipe.

### **Procedure GET\_RECIPE\_ID**

This procedure, based on validity rule ID, returns the recipe ID associated with this validity rule.

### **Procedure GET\_RECIPE\_STEP\_DETAILS**

This procedure, based on recipe ID, returns the recipe routings step details associated with this recipe.

### **Procedure GET\_ROUTING\_ID**

This procedure, based on recipe ID or recipe number and version, returns the routing ID associated with this recipe.

### **Procedure GET\_ROUTING\_STEP\_DETAILS**

This procedure gets the routing step information specific to the recipe.

**Procedure GET\_ROUT\_HDR**

This procedure, based on recipe ID, returns the routing details associated with this recipe.

**Procedure GET\_ROUT\_MATERIAL**

This procedure, based on recipe ID, returns the steps material association for this recipe.

**Procedure GET\_STEP\_DEPD\_DETAILS**

This procedure, based on recipe ID, returns the recipe routing step dependency information.

## Recipe Validity Rule Fetch

API to fetch validity rules specific to a given recipe. This API can also be called by providing the total output quantity or total input quantity. This API is called extensively by other applications, such as Process Execution, Process Planning and Costing.

**Procedure GET\_BATCHFORMULA\_RATIO**

This procedure is responsible for determining the ratio of the batch input quantity to the formula input quantity while determining validity rules based on total input quantity.

**Procedure GET\_CONTRIBUTING\_QTY**

This procedure is responsible for determining the actual contributing quantity of the formula.

**Procedure GET\_INGREDPROD\_RATIO**

This procedure is responsible for determining the ratio of the products to ingredients while trying to determine validity rules based on total input quantity.

**Procedure GET\_INPUT\_RATIO**

This procedure is responsible for determining the actual ratio of product for the total input quantity.

**Procedure GET\_OUTPUT\_RATIO**

This procedure is responsible for determining the output ratio, which is the ratio of the batch output to the formula output, when a total output quantity is used as the criteria for a validity rule.

**Procedure GET\_VALIDITY\_RULES**

This PL/SQL procedure is responsible for getting the validity rules based on the input parameters.

**Procedure UOM\_CONVERSION\_MESG**

This procedure indicates that the unit of measure conversion occurred successfully.

## Routing

A routing is a sequenced set of operations, organized in steps, that must be performed in order to complete a production batch. A routing must include at least one routing step (operation).

### Routing Header Table Type Attributes

The Routing Header contains information applicable to entire routing.

Field/Column	Type	Length	Default	Required	Validation
ROUTING_ID	number	10	n/a	Y	Routing ID.
ROUTING_NO	varchar2	32	n/a	Y	The routing number and routing version must be unique.
ROUTING_VER	number	5	n/a	Y	The routing number and routing version must be unique. The lowest value is 0, the highest value is 99999.
ROUTING_QTY	number	variable	n/a	Y	The routing quantity must be greater than or equal to 0.
ITEM_UM	varchar2	4	n/a	Y	Valid values for the item unit of measure are from sy_uoms_mst where the delete_mark is equal to 0.
ROUTING_DESC	varchar2	40	n/a	Y	Routing description.
DELETE_MARK	number	5	0	Y	Valid values for delete mark are 0 = active, 1 = inactive.
OWNER_ORGN_CODE	varchar2	4	user orgn_code	N	The valid values for owner organization code are from sy_orgn_usr and sy_orgn_mst.

Field/Column	Type	Length	Default	Required	Validation
ROUTING_CLASS	varchar2	4	null	N	Valid values for routing class are from the fm_rout_cls table where the delete_mark is equal to 0
EFFECTIVE_START_DATE	date	variable	sysdate	N	The effective start date cannot be greater than the effective end date.
EFFECTIVE_END_DATE	date	variable	null	N	The effective end date must not be greater than any operation to date.
OWNER_ID	number	15	fn_profile.VALUE('USER_ID');	N	Owner ID.
PROCESS_LOSS	number	variable	null	N	Process loss.
ROUTING_STATUS	varchar2	30	100	N	Routing status.
ENFORCE_STEP_DEPENDENCY	number	5	0	N	If this is set to Yes, then fm_rout_dtl.steprelease type is set to manual.
ATTRIBUTE1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE7	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE10	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE11	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE12	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE13	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE14	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE15	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE16	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE18	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE19	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE20	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE21	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE22	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE23	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE24	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE25	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE26	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE27	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE28	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE29	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE30	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE_CATEGORY	varchar2	30	null	N	Descriptive Flexfield Segment.

### Procedure INSERT\_ROUTING

This function can be used to add routing information. It generates the routing\_id, which is the primary key to the gmd\_routings table, and routingstep\_id, which is the primary key to the fm\_rout\_dtl table.

This function calls the insert\_routing\_step function. There must be at least one routing step for a routing header to be inserted. Optionally, the insert\_step\_dependency function can be called creating routing step dependencies for routing steps inserted.

### Procedure UPDATE\_ROUTING

This function lets you update a routing. The gmd\_routings.routing\_id or routing\_no and routing\_vers must be passed in. The column to be updated and the value of that column must be passed to the function.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

### Procedure DELETE\_ROUTING

The gmd\_routings.routing\_id or gmd\_routings.routing\_no and routing\_vers uniquely identifies a routing and must be passed to the API so the routing can be

deleted. This function then calls the `update_routing` function and sets the `delete_mark` to 1.

If a routing is used in a recipe or batch, then it cannot be deleted.

### **Procedure UNDELETE\_ROUTING**

The `gmd_routings.routing_id` or `gmd_routings.routing_no` and `routing_vers` uniquely identifies a routing and must be passed to the API so the routing can be undeleted. This function then calls the `update_routing` function and sets the `delete_mark` to 0.

## Routing Steps

A routing is a sequenced set of operations, organized in steps, that must be performed in order to complete a production batch. A routing must include at least one routing step (operation).

### Routing Detail Table Type Attributes

The Routing Details contains the operations that comprise a routing and their scalability types.

Field/Column	Type	Length	Default	Required	Validation
ROUTING_ID	number	10	n/a	Y	The routing ID and routingstep number must unique.
ROUTINGSTEP_NO	number	5	n/a	Y	The routing ID and routingstep number must unique.
ROUTINGSTEP_ID	number	10	n/a	Y	Routingstep ID.
OPRN_ID	number	10	n/a	Y	The valid values for operation ID are from gmd_operations.
STEP_QTY	number	variable	0	Y	The step quantity cannot be less than 0.
STEPRELEASE_TYPE	number	5	1	Y	Valid values for steprelease type are 0 = automatic, 1 = manual. If gmd_routings.enforce_step_dependency is set to Yes, then the steprelease_type is set to manual.
ATTRIBUTE1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE4	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE10	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE11	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE12	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE13	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE14	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE15	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE16	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE18	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE19	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE20	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE2 1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3 0	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE_ CATEGORY	varchar2	30	null	N	Descriptive Flexfield Segment.

### Procedure INSERT\_ROUTING\_STEPS

This function is used to add routing detail information to the fm\_rout\_dtl table. It requires the gmd\_routings.routing\_id or gmd\_routings.routing\_no and routing\_vers to be passed in for the routing being updated. It generates the routingstep\_id, which is the unique key to the fm\_rout\_dtl table. The routing\_id and routingstep\_no used must be unique as they define the primary key for the fm\_rout\_dtl table. You can add routing step dependencies to the routingstep being added.

### Procedure UPDATE\_ROUTING\_STEPS

This function lets you update a routing step record. It requires the fm\_rout\_dtl.routingstep\_id, or routingstep\_no and routing\_id (or routing\_no and routing\_vers) to be passed in. In addition, a column to be updated and the value of that column must be passed to the function.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

### **Procedure DELETE\_ROUTING\_STEPS**

The fm\_rout\_dtl.routingstep\_id or routingstep\_no and routing\_id, or routingstep\_no, routing\_no, and routing\_vers uniquely identifies a routing step. It must be passed to the API so the routing step can be deleted.

---

---

**Note:** If there are any step material associations existing, or if the step quantity has been overridden at the recipe level, then the deletion of the routing step is not allowed.

---

---

This function calls the delete\_step\_dependency function.

## Routing Step Dependency

Routing step dependency information can be set up enabling you to create and modify relationships between routing steps. This information includes previous step, dependency type, standard delay, and transfer percent data.

### Routing Step Dependency Table Type Attributes

The Routing step dependencies defines relationships from one step to another. such as sequences, delays, and transfer quantities.

Field/Column	Type	Length	Default	Required	Validation
ROUTINGST EP_NO	number	5	n/a	Y	The routingstep number, dependency routingstep number, and routing ID must be unique.
DEP_ ROUTINGST EP_NO	number	5	n/a	Y	The routingstep number, dependency routingstep number, and routing ID must be unique.
ROUTING_ID	number	10	n/a	Y	The routingstep number, dependency routingstep number, and routing ID must be unique.
DEP_TYPE	number	5	start to start	Y	The valid values for dependency type are start to start, and finish to finish.
REWORK_ CODE	varchar2	5	null	N	Rework code.
TRANSFER_ QTY	number	variable	0	Y	The lowest transfer quantity allowed is 0.
ITEM_UM	varchar2	4	n/a	Y	Item unit of measure.
TRANSFER_ PCT	number	variable	100	N	The transfer percentage must be less than or equal to 100 and greater than or equal to 0. If it is null, then it defaults to 0.

### **Procedure INSERT\_STEP\_DEPENDENCY**

This function is used to add routing step dependency information for routing steps. It requires:

- `fm_rout_dtl.routingstep_no` and `routing_id`, or
- `fm_rout_dtl.routingstep_id`, or
- `routingstep_no` and `routing_no` and `routing_vers`

to be passed in for the routingstep being updated. The `routingstep_no`, `dep_routingstep_no`, and the `routing_id` entered must be unique as they define the primary key for the `fm_rout_dep` table.

### **Procedure UPDATE\_STEP\_DEPENDENCY**

This function lets you update a routing step record. It requires the `fm_rout_dep.routingstep_no`, `dep_routingstep_no`, and `routing_id`, or `fm_rout_dtl.routing_no` and `routing_vers` to be passed in. In addition, the column to be updated and the value of that column must be passed to the function.

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

### **Procedure DELETE\_STEP\_DEPENDENCY**

The `fm_rout_dep.routing_id` (or `gmd_routings.routing_no` and `routing_vers`) and `routingstep_no`, `dep_routingstep_no` uniquely identify a routingstep dependency record. They must be passed to the API so the step dependency can be deleted.

## Operation

The Operation APIs are business objects that can insert, update, or delete operation information.

### Operation Header Table Type Attributes

Field/Column	Type	Length	Default	Required	Validation
OPRN_ID	number	10	n/a	Y	The operation ID is the surrogate for oprn_no and oprn_vers.
OPRN_NO	varchar2	16	n/a	Y	Operation number and operation version must be unique.
OPRN_VERS	number	5	n/a	Y	Operation number and operation version must be unique. The value must be between 0 and 99999.
OPRN_DESC	varchar2	40	n/a	Y	Operation description.
PROCESS_QTY_UM	varchar2	4	n/a	Y	The process quantity unit of measure must be a valid unit of measure from sy_uoms_mst where the delete_mark is equal to 0.
OPRN_CLASS	varchar2	4	n/a	N	The operation class must be valid from the gmd_operation_class_b/_tl table.
POC_CTL_CLASS	varchar2	8	n/a	N	Not used.
EFFECTIVE_START_DATE	date	variable	sysdate	Y	The effective start date must be less than the effective end date.
EFFECTIVE_END_DATE	date	variable	n/a	N	The effective end date must be greater than the effective start date.
OWNER_ORGN_CODE	varchar2	4	See Validation	Y	If new, then the default value is derived from FND_PROFILE.VALUE('GEMMS_DEFAULT_ORGN');.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE10	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE11	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE12	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE13	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE14	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE15	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE16	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE18	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE19	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE20	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE21	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE22	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE23	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE24	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE25	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE26	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE27	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE28	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE29	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE30	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE_CATEGORY	varchar2	30	null	N	Descriptive Flexfield Segment.

### Procedure INSERT\_OPERATION

This function can be used to add operation information. It generates the oprn\_id, which is the primary key to the gmd\_operations table, and oprn\_line\_id, which is the primary key to the gmd\_operation\_activities table.

This function calls the `insert_operation_activity` function.

`Gmd_operations.operation_status` is defaulted to 100 (New) and cannot be changed or passed in. `Gmd_operations.delete_mark` is defaulted to 0 (active) and cannot be changed or passed in.

### **Procedure UPDATE\_OPERATION**

This function lets you update an operation. The `oprn_id` or `oprn_no` and `oprn_vers` must be passed in. The column to be updated and the value of that column must be passed to the function.

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

### **Procedure DELETE\_OPERATION**

The `gmd_operations.oprn_id` or `gmd_operations.oprn_no` and `gmd_operations.oprn_vers` uniquely identifies an operation and must be passed to the API so the operation can be deleted.

This function calls the `gmd_operation.update_operation` function and sets `delete_mark` to 1. It also calls the `gmd_operation.update_operation_activity` and `gmd_operation.update_operation_resource` functions and set the `delete_mark` for the associated operation activities and resources to 1.

If an operation is used in a recipe or batch, then it cannot be deleted.

## Operation Activity

The Operation Activity APIs are business objects that can insert, update, or delete operation activity information.

### Operation Activity Header Table Type Attributes

Field/Column	Type	Length	Default	Required	Validation
OPRN_LINE_ID	number	15	n/a	Y	Operation line ID.
OPRN_ID	number	15	n/a	Y	The operation ID is the surrogate for oprn_no and oprn_vers.
ACTIVITY	varchar2	16	n/a	Y	This must be a valid activity in FM_ACTV_MST.
OFFSET_INTERVAL	number	variable	n/a	Y	The format for offset interval is 99999D9999.
ACTIVITY_FACTOR	number	variable	1	Y	Activity factor.
SEQUENCE_DEPENDENT_IND	varchar2	4	0	N	Sequence dependent indicator.
ATTRIBUTE1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE7	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE10	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE11	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE12	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE13	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE14	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE15	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE16	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE17	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE18	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE19	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE20	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE21	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE22	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE23	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE2 4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3 0	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE_ CATEGORY	varchar2	30	null	N	Descriptive Flexfield Segment.

### Procedure INSERT\_OPERATION\_ACTIVITY

This function is used to add operation detail information to the `gmd_operation_activities` and `gmd_operation_resources` table. It requires the `gmd_operation.oprn_id` or `gmd_operation.oprn_no` and `gmd_operation.oprn_vers` to be passed in from the operation being updated. It generates the `oprn_line_id`, which is the primary key to the `gmd_operation_activities`. You can add multiple operation resources to this activity.

This function is called by the `gmd_operations.insert_operation` function, and in turn calls the `insert_operation_resources` function.

The `gmd_operations_activities.delete_mark` is defaulted to 0 (active) and cannot be changed or passed in.

### Procedure UPDATE\_OPERATION\_ACTIVITY

This function lets you update an operation activity record. The `oprn_line_id`, the column to be updated, and the value of that column must be passed to the function.

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

**Procedure DELETE\_OPERATION\_ACTIVITY**

The `gmd_operations_activities.oprn_line_id` uniquely identifies an operation activity. It must be passed to the API so the operation activity can be deleted.

## Operation Resources

The Operation Resources APIs are business objects that can insert, update, or delete operation resources information.

### Operation Resources Header Table Type Attributes

Field/Column	Type	Length	Default	Required	Validation
OPRN_LINE_ID	number	15	n/a	Y	Operation line ID.
RESOURCES	varchar2	16	n/a	Y	The resources must be active and exist in CR_RSRC_MST.
RESOURCE_USAGE	number	variable	n/a	Y	The resource usage may have a lowest value of 0.
RESOURCE_COUNT	number	variable	1	Y	The resource count values are low value of 0 and high value of 99.
USAGE_UM	varchar2	4	n/a	Y	The usage unit of measure must exist in sy_uoms_mst.
PROCESS_QTY	number	variable	n/a	Y	The lowest process quantity value is 0.
PROCESS_UOM	varchar2	4	n/a	Y	The process unit of measure comes from gmd_operations process_qty_um.
PRIN_RSRC_IND	number	variable	n/a	Y	If operation resources exist, then there must be at least one row in the table where this is set to 1. It must have only one row in the table where this is set to 1.
SCALE_TYPE	number	5	1	Y	Scale type.
COST_ANALYSIS_CODE	varchar2	4	n/a	Y	Cost analysis code.

Field/Column	Type	Length	Default	Required	Validation
COST_CMPNTCLS_ID	number	10	n/a	Y	Cost component class ID.
OFFSET_INTERVAL	number	variable	0	Y	The lower offset interval value is 0.
MIN_CAPACITY	number	variable	null	N	Minimum capacity.
MAX_CAPACITY	number	variable	null	N	Maximum capacity.
CAPACITY_UOM	varchar2	4	null	N	Capacity unit of measure.
ATTRIBUTE1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE10	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE11	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE12	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE1 3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 8	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE1 9	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 0	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 1	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 2	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 3	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 4	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 5	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 6	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 7	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE2 8	varchar2	240	null	N	Descriptive Flexfield Segment.

Field/Column	Type	Length	Default	Required	Validation
ATTRIBUTE29	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE30	varchar2	240	null	N	Descriptive Flexfield Segment.
ATTRIBUTE_CATEGORY	varchar2	30	null	N	Descriptive Flexfield Segment.

### **Procedure INSERT\_OPERATION\_RESOURCES**

This function is used to add operation resource information for operation activities. It requires that `gmd_operation_activities.oprn_line_id` be passed in for the operation activity being updated. The `oprn_line_id` and resource entered must be unique as they define the primary key for the `gmd_operation_resources` table.

The `gmd_operation_resources.delete_mark` is defaulted to 0 (active) and cannot be changed or passed in.

### **Procedure UPDATE\_OPERATION\_RESOURCES**

This function lets you update an operation resource record. The `oprn_line_id`, resource, the column to be updated, and the value of that column must be passed to the function.

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

### **Procedure DELETE\_OPERATION\_RESOURCES**

The `gmd_operation_resources.oprn_line_id` and `gmd_operation_resources.resource` uniquely identify an operation resource. They must be passed to the API so the operation resource can be deleted.

## Activity

The Activity APIs are business objects that can insert, update, or delete activity information.

### Activity Header Table Type Attributes

Field/Column	Type	Length	Default	Required	Validation
ACTIVITY	varchar2	16	n/a	Y	Activity.
COST_ANALYSIS_CODE	varchar2	4	n/a	Y	The cost analysis code selects from cm_aly_s_mst where the delete_mark is equal to 0.
DELETE_MARK	number	5	0	Y	Valid values for delete mark are 0 = active, 1 = inactive.
TRANS_CNT	number	10	n/a	N	Transaction count.
ACTIVITY_DESC	varchar2	40	n/a	Y	Activity description.

### Procedure INSERT\_ACTIVITY

This function can be used to add an activity to the gmd\_activities table. Multiple activities can be added at once using this function.

### Procedure UPDATE\_ACTIVITY

This function can be used to modify an activity in the gmd\_activities table. The activity must be passed in. The column to be updated and the value of that column must be passed to the function.

If any column must be updated to NULL, then you must pass in FND\_API.G\_MISS\_CHAR, FND\_API.G\_MISS\_NUM, or FND\_API.G\_MISS\_DATE variables to the API column values to update the column to NULL in the database.

### Procedure DELETE\_ACTIVITY

This function can be used to delete an activity in the gmd\_activities table. The activity must be passed in. This function calls the gmd\_activities.update\_activity function and sets the delete\_mark to 1.



# A

---

---

## Messages and Errors

This appendix covers the following topics:

- 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, then it also returns the message data.

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

To add a message to the API message list, developers 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 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 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 better performance and reduction in the overall number of calls a program needs to make in order to execute an API, it is recommended that APIs provide their callers with the following information:

- message count
- message data

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

```
x_msg_count          OUT          NUMBER
x_msg_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 for success
- E for error
- U for unknown or unexpected status

## Understanding Error Messages

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

### Displaying Errors in Languages Other than English

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

The following is a complete listing of Recipe 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 is populated at runtime.

Message Name	Message Code
Cost Analysis code is undefined	COST_ANALYS_CODE_UNDEFINED
Duplicate Recipes are not allowed	GMD_DUP_RECIPE
Invalid Cost Analysis Code	GMD_INVALID_COST_ANALYS_CODE
Invalid Cost Component Class ID	GMD_INVALID_COST_CMPNTCLS_ID
Invalid Operations Line ID	GMD_INVALID_OPRNLINE_ID
Operation can not be updated with status of On Hold or Obsolete/Archived or if Marked For Purge	GMD_OPRN_NOT_VALID
Recipe needs a valid customer	GMD_RECIPE_CUSTOMER_INVALID

<b>Message Name</b>	<b>Message Code</b>
Recipe Description is either missing or too long	GMD_RECIPE_DESCRIPTION
Recipe has an invalid number or version	GMD_RECIPE_NOT_VALID
Recipe Organization is invalid	GMD_RECIPE_ORGN_INVALID
Routing used for this recipe is invalid	GMD_RECIPE_ROUTING_INVALID
Transaction complete records applied and saved	GMD_SAVED_CHANGES
Unexpected Error	GMD_UNEXPECTED_ERROR
Invalid Min/Max Date	QC_MIN_MAX_DATE
Invalid Unit of Measure Code	SY_INVALID_UM_CODE
Field is Required	SY_REQUIRED

---

---

## How to Get Your OPM Recipe APIs Running

This appendix is used in conjunction with the rest of the guide, but is not intended to replace it. The information below is supplemental material that provides additional information regarding OPM Recipe APIs.

Following are four steps to use the Recipe APIs:

1. Creating a Wrapper file for Recipe Header
2. Running the Recipe Header Wrapper file
3. Creating a Wrapper file for Recipe Details
4. Running the Recipe Detail Wrapper file

### Step 1 Creating the Wrapper File for Recipe Header

The Recipe Header APIs expects the recipe data to be passed using a PL/SQL table (one of the parameters for the API). In constructing the PL/SQL table, it is important to assign all required values to the columns in this table.

#### Recipe Header Wrapper File Example

This example details the contents from the Recipe Header Wrapper file. The filename is `test_recipe_hdr.sql`. In this file, a Recipe 'SS-SPECIAL' version 1 is created.

```
set serveroutput on
set echo off
set verify off

Declare
    l_return_statusvarchar2(1);
    l_msg_datavarchar2(240);
    l_msg_countnumber;
```

---

```

recipe_hdr_tbl gmd_recipe_header.recipe_tbl;
recipe_up_flexgmd_recipe_header.recipe_update_flex;
recipe_flex_tblgmd_recipe_header.recipe_flex;

my_counternumber := 1;

Begin

recipe_hdr_tbl(my_counter).RECIPE_NO           := 'SS-SPECIAL'   ;
recipe_hdr_tbl(my_counter).RECIPE_VERSION      := 1                  ;
recipe_hdr_tbl(my_counter).RECIPE_DESCRIPTION := 'Created this recipe thro
API';
recipe_hdr_tbl(my_counter).RECIPE_STATUS       := '700'          ;

recipe_hdr_tbl(my_counter).FORMULA_NO         := 'SHY-FORM'      ;
recipe_hdr_tbl(my_counter).FORMULA_VERS      := 4                  ;
recipe_hdr_tbl(my_counter).ROUTING_NO        := 'SHY-RT-CHARGES' ;
recipe_hdr_tbl(my_counter).ROUTING_VERS     := 1                  ;

recipe_hdr_tbl(my_counter).DELETE_MARK       := 0                  ;
recipe_hdr_tbl(my_counter).CREATION_DATE     := SYSDATE          ;
recipe_hdr_tbl(my_counter).CREATED_BY        := 2060             ;
recipe_hdr_tbl(my_counter).LAST_UPDATED_BY   := 2060             ;
recipe_hdr_tbl(my_counter).LAST_UPDATE_DATE := SYSDATE          ;
recipe_hdr_tbl(my_counter).LAST_UPDATE_LOGIN := 2060             ;

recipe_hdr_tbl(my_counter).USER_NAME:= 'ORAUSR'      ;
recipe_hdr_tbl(my_counter).OWNER_ORGN_CODE  := 'OPME'         ;
recipe_hdr_tbl(my_counter).CREATION_ORGN_CODE := 'OPME'         ;
recipe_hdr_tbl(my_counter).OWNER_ID         := 2060             ;

recipe_flex_tbl(my_counter).attributel := 'FLEX1';

GMD_RECIPE_HEADER.CREATE_RECIPE_HEADER
(p_api_version=> 1.0,
p_init_msg_list=> FND_API.G_FALSE ,
p_commit=> FND_API.G_FALSE ,
p_called_from_forms=> 'NO',
x_return_status=> l_return_status,
x_msg_count=> l_msg_count,
x_msg_data=> l_msg_data,
p_recipe_header_tbl => recipe_hdr_tbl,
p_recipe_header_flex=> recipe_flex_tbl
);

```

---

```
end;  
/  
show errors;
```

## Step 2 Running the Recipe Header Wrapper File

The Recipe APIs are PL/SQL based and run database packages, and are run by SQL\*Plus or Unix, initiating a sql session first.

### Via Unix

```
sqlplus apps/apps@database @test_recipe_hdr.sql
```

### Via Sql\*Plus

Login and start the wrapper file using:

```
@test_recipe_hdr.sql
```

## Step 3 Creating the Wrapper File for Recipe Details

This example creates a Recipe Validity Rule. The Recipe Detail API expects the Recipe Header to be created and the detail data to be passed using a PL/SQL table (one of the parameters for the API). In constructing the PL/SQL table, it is important to assign all required values to the columns in this table.

### Recipe Detail Wrapper File Example

The example details the contents from the Recipe Detail - Validity Rule Wrapper file. In this example the filename is named test\_recipe\_dtl.sql. In this file, a Validity Rule for Recipe - 'SS-SPECIAL' version 1 is created.

```
set serveroutput on  
declare  
  l_return_status      varchar2(1);  
  l_msg_data           varchar2(240);  
  l_msg_count          number;  
  -- table definition  
  recipe_dtl_tbl      gmd_recipe_detail.recipe_detail_tbl;  
  vr_tbl              gmd_recipe_detail.recipe_vr_tbl;  
  mtl_tbl             gmd_recipe_detail.recipe_mtl_tbl;  
  
  -- flex field definition  
  recipe_flex_tbl     gmd_recipe_detail.recipe_flex;  
  recipe_up_flex      gmd_recipe_detail.recipe_update_flex;
```

```

my_counter          number := 1;
create_type         VARCHAR2(30) := 'VR';

begin

/*  -- ***** Recipe header info
recipe_dtl_tbl(my_counter).RECIPE_NO           := 'SS_SPECIAL';
recipe_dtl_tbl(my_counter).RECIPE_VERSION     := 1;
recipe_dtl_tbl(my_counter).USER_NAME         := 'ORAUSR'
-- *****process loss related
recipe_dtl_tbl(my_counter).ORGN_CODE         := 'OPME';
recipe_dtl_tbl(my_counter).PROCESS_LOSS      := 8;
-- *****customer related
recipe_dtl_tbl(my_counter).CUSTOMER_NO       := 'Oracle Inventory';
-- *****routing step related
recipe_dtl_tbl(my_counter).ROUTINGSTEP_ID    := 459;
recipe_dtl_tbl(my_counter).STEP_QTY         := 300;
-- *****orgn activity related
recipe_dtl_tbl(my_counter).oprn_line_id      := 44;
recipe_dtl_tbl(my_counter).activity_factor   := 3;
-- *****resource related
recipe_dtl_tbl(my_counter).resources        := 'CN-DRYER';
recipe_dtl_tbl(my_counter).oprn_line_id     := 44;
recipe_dtl_tbl(my_counter).min_capacity     := 45;
recipe_dtl_tbl(my_counter).max_capacity     := 105;
recipe_dtl_tbl(my_counter).capacity_uom    := 'KGM';
recipe_dtl_tbl(my_counter).DELETE_MARK     := 0;
recipe_dtl_tbl(my_counter).CREATION_DATE    := SYSDATE;
recipe_dtl_tbl(my_counter).CREATED_BY      := 2060;
recipe_dtl_tbl(my_counter).LAST_UPDATED_BY := 2060;
recipe_dtl_tbl(my_counter).LAST_UPDATE_DATE := SYSDATE;
recipe_dtl_tbl(my_counter).LAST_UPDATE_LOGIN := 2060;
*/

/*
-- *****Creating RECIPE MATERIAL STEP LINE related
mtl_tbl(my_counter).RECIPE_NO           := 'SS_SPECIAL';
mtl_tbl(my_counter).RECIPE_VERSION     := 1;
mtl_tbl(my_counter).USER_NAME         := 'ORAUSR';
mtl_tbl(my_counter).FORMULALINE_ID    := 1906;
mtl_tbl(my_counter).CREATED_BY      := 2060;
mtl_tbl(my_counter).CREATION_DATE    := SYSDATE;
mtl_tbl(my_counter).LAST_UPDATED_BY := 2060;
*/

```

```

mtl_tbl(my_counter).LAST_UPDATE_DATE      := SYSDATE;
mtl_tbl(my_counter).LAST_UPDATE_LOGIN    := 2060;
mtl_tbl(my_counter).ROUTINGSTEP_ID       := 459;
*/

-- *****Creating VALIDITY RULE RELATED
vr_tbl(my_counter).RECIPE_NO              := 'SS_SPECIAL';
vr_tbl(my_counter).RECIPE_VERSION         := 1;
vr_tbl(my_counter).USER_NAME              := 'ORAUSR';
vr_tbl(my_counter).ORGN_CODE              := 'OPME';
vr_tbl(my_counter).ITEM_NO                := 'ABH-PROD2';
vr_tbl(my_counter).ITEM_UM                := 'KGM';

vr_tbl(my_counter).RECIPE_USE              := 0;
vr_tbl(my_counter).PREFERENCE              := 1;
vr_tbl(my_counter).START_DATE              := SYSDATE;
vr_tbl(my_counter).END_DATE                := '24-FEB-2003';
vr_tbl(my_counter).MIN_QTY                 := 100;
vr_tbl(my_counter).MAX_QTY                 := 500;
vr_tbl(my_counter).STD_QTY                 := 500;
vr_tbl(my_counter).INV_MIN_QTY             := 100;
vr_tbl(my_counter).INV_MAX_QTY             := 1000;
vr_tbl(my_counter).CREATED_BY              := 2060;
vr_tbl(my_counter).CREATION_DATE           := SYSDATE;
vr_tbl(my_counter).LAST_UPDATED_BY        := 2060;
vr_tbl(my_counter).LAST_UPDATE_DATE       := SYSDATE;
vr_tbl(my_counter).LAST_UPDATE_LOGIN      := 2060;
vr_tbl(my_counter).DELETE_MARK             := 0;
vr_tbl(my_counter).VALIDITY_RULE_STATUS    := '700';

If (create_type = 'LOSS') then
  GMD_RECIPE_DETAIL.CREATE_RECIPE_PROCESS_LOSS
  (
    p_api_version          => 1.0
    ,
    p_init_msg_list        => FND_API.G_FALSE
    ,
    p_commit                => FND_API.G_FALSE
    ,
    p_called_from_forms    => 'NO'
    ,
    x_return_status        => l_return_status
    ,
    x_msg_count            => l_msg_count
    ,
    x_msg_data             => l_msg_data
    ,
    p_recipe_detail_tbl    => recipe_dtl_tbl
  );

elseif (create_type = 'CUST') then

```

```

GMD_RECIPE_DETAIL.CREATE_RECIPE_CUSTOMERS
(
  p_api_version          => 1.0
  p_init_msg_list       => FND_API.G_FALSE
  p_commit               => FND_API.G_FALSE
  p_called_from_forms   => 'NO'
  x_return_status       => l_return_status
  x_msg_count           => l_msg_count
  x_msg_data            => l_msg_data
  p_recipe_detail_tbl   => recipe_dtl_tbl
);

elsif (create_type = 'VR') then

  GMD_RECIPE_DETAIL.CREATE_RECIPE_VR
(
  p_api_version          => 1.0
  p_init_msg_list       => FND_API.G_FALSE
  p_commit               => FND_API.G_FALSE
  p_called_from_forms   => 'NO'
  x_return_status       => l_return_status
  x_msg_count           => l_msg_count
  x_msg_data            => l_msg_data
  p_recipe_vr_tbl       => vr_tbl
  p_recipe_vr_flex      => recipe_flex_tbl
);

elsif (create_type = 'MTL') then

  GMD_RECIPE_DETAIL.CREATE_RECIPE_MTL
(
  p_api_version          => 1.0
  p_init_msg_list       => FND_API.G_FALSE
  p_commit               => FND_API.G_FALSE
  p_called_from_forms   => 'NO'
  x_return_status       => l_return_status
  x_msg_count           => l_msg_count
  x_msg_data            => l_msg_data
  p_recipe_mtl_tbl     => mtl_tbl
);

elsif (create_type = 'STEP') THEN

  GMD_RECIPE_DETAIL.RECIPE_ROUTING_STEPS
(
  p_api_version          => 1.0
  p_init_msg_list       => FND_API.G_FALSE
  p_commit               => FND_API.G_FALSE
  p_called_from_forms   => 'NO'

```

```

        x_return_status      => l_return_status      ,
        x_msg_count         => l_msg_count          ,
        x_msg_data          => l_msg_data           ,
        p_recipe_detail_tbl => recipe_dtl_tbl       ,
        p_recipe_insert_flex => recipe_flex_tbl      ,
        p_recipe_update_flex => recipe_up_flex
    );

    elsif (create_type = 'ACT') THEN

        GMD_RECIPE_DETAIL.RECIPE_ORGN_OPERATIONS
    (
        p_api_version      => 1.0                    ,
        p_init_msg_list    => FND_API.G_FALSE        ,
        p_commit           => FND_API.G_FALSE        ,
        p_called_from_forms => 'NO'                  ,
        x_return_status    => l_return_status        ,
        x_msg_count        => l_msg_count            ,
        x_msg_data         => l_msg_data             ,
        p_recipe_detail_tbl => recipe_dtl_tbl       ,
        p_recipe_insert_flex => recipe_flex_tbl      ,
        p_recipe_update_flex => recipe_up_flex
    );

    elsif (create_type = 'RES') THEN

        GMD_RECIPE_DETAIL.RECIPE_ORGN_RESOURCES
    (
        p_api_version      => 1.0                    ,
        p_init_msg_list    => FND_API.G_FALSE        ,
        p_commit           => FND_API.G_FALSE        ,
        p_called_from_forms => 'NO'                  ,
        x_return_status    => l_return_status        ,
        x_msg_count        => l_msg_count            ,
        x_msg_data         => l_msg_data             ,
        p_recipe_detail_tbl => recipe_dtl_tbl       ,
        p_recipe_insert_flex => recipe_flex_tbl      ,
        p_recipe_update_flex => recipe_up_flex
    );

    end if;

    dbms_output.put_line('the status is '||l_return_status);

end;
/
show errors;

```

---

#### **Step 4 Running the Recipe Detail Wrapper File**

The Recipe APIs are PL/SQL based and run database packages, and are run by SQL\*Plus or Unix, initiating a sql session first.

##### **Via Unix**

```
sqlplus apps/apps@database @test_recipe_dtl.sql
```

##### **Via Sql\*Plus**

Login and start the wrapper file using:

```
@test_recipe_dtl.sql
```

---

---

# Index

## B

---

Batch File, 1-4

## C

---

COST\_ANALYS\_CODE\_UNDEFINED, A-5  
CREATE\_RECIPE\_CUSTOMERS, 4-14  
CREATE\_RECIPE\_HEADER, 4-7  
CREATE\_RECIPE\_MTL, 4-15  
CREATE\_RECIPE\_PROCESS\_LOSS, 4-16  
CREATE\_RECIPE\_VR, 4-16

## D

---

DELETE\_ACTIVITY, 4-47  
DELETE\_OPERATION, 4-38  
DELETE\_OPERATION\_ACTIVITY, 4-42  
DELETE\_OPERATION\_RESOURCES, 4-46  
DELETE\_RECIPE\_HEADER, 4-7  
DELETE\_ROUTING, 4-27  
DELETE\_ROUTING\_STEPS, 4-32  
DELETE\_STEP\_DEPENDENCY, 4-34

## E

---

error, 1-2  
Error Messages, A-5

## F

---

file, 3-7, 3-8, 3-9, 3-10, 3-11, 3-12, 3-13, 3-14  
files, 3-2, 3-3, 3-5, 3-6  
FND\_API, 1-3

FND\_MESSAGE, 1-3  
FND\_PUB\_MSG, 1-3  
FND\_USR, 2-3

## G

---

GET\_MESSAGES, 1-6  
GET\_BATCHFORMULA\_RATIO, 4-22  
GET\_CONTRIBUTING\_QTY, 4-22  
GET\_FORMULA\_ID, 4-21  
GET\_INGREDPROD\_RATIO, 4-22  
GET\_INPUT\_RATIO, 4-22  
GET\_OPRN\_ACT\_DETL, 4-21  
GET\_OPRN\_RESC\_DETL, 4-21  
GET\_OUTPUT\_RATIO, 4-23  
GET\_PROCESS\_LOSS, 4-21  
GET\_RECIPE\_ID, 4-21  
GET\_RECIPE\_STEP\_DETAILS, 4-21  
GET\_ROUT\_HDR, 4-22  
GET\_ROUT\_MATERIAL, 4-22  
GET\_ROUTING\_ID, 4-21  
GET\_ROUTING\_STEP\_DETAILS, 4-21  
GET\_STEP\_DEPD\_DETAILS, 4-22  
GET\_VALIDITY\_RULES, 4-23  
GMD\_ACTIVITY\_PUB, 1-9  
GMD\_DUP\_RECIPE, A-5  
GMD\_FORMULA\_DETAIL\_PUB, 1-8  
GMD\_FORMULA\_DETAIL\_PVT, 1-8  
GMD\_FORMULA\_EFFECTIVITY\_PUB, 1-8  
GMD\_FORMULA\_PUB, 1-8  
GMD\_FORMULA\_PVT, 1-8  
GMD\_INVALID\_COST\_ANALYS\_CODE, A-5  
GMD\_INVALID\_COST\_CMPNTCLS\_ID, A-5  
GMD\_INVALID\_OPRNLINE\_ID, A-5

GMD\_OPERATION\_ACTIVITIES\_PUB, 1-9  
 GMD\_OPERATION\_PUB, 1-9  
 GMD\_OPERATION\_RESOURCES\_PUB, 1-9  
 GMD\_OPRN\_NOT\_VALID, A-5  
 GMD\_RECIPE\_CUSTOMER\_INVALID, A-5  
 GMD\_RECIPE\_DESCRIPTION, A-6  
 GMD\_RECIPE\_NOT\_VALID, A-6  
 GMD\_RECIPE\_ORGN\_INVALID, A-6  
 GMD\_RECIPE\_ROUTING\_INVALID, A-6  
 GMD\_ROUTING\_STEPS\_PUB, 1-8  
 GMD\_ROUTINGS\_PUB, 1-8  
 GMD\_SAVED\_CHANGES, A-6  
 GMD\_STEP\_DEPENDENCY\_PUB, 1-8  
 GMD\_UNEXPECTED\_ERROR, A-6  
 GMDPACTB.pls, 1-9  
 GMDPACTS.pls, 1-9  
 GMDPFMDB.pls, 1-8  
 GMDPFMDS.pls, 1-8  
 GMDPFMEB.pls, 1-8  
 GMDPFMES.pls, 1-8  
 GMDPFMHB.pls, 1-8  
 GMDPFMHS.pls, 1-8  
 GMDPOPAB.pls, 1-9  
 GMDPOPAS.pls, 1-9  
 GMDPOPNB.pls, 1-9  
 GMDPOPNS.pls, 1-9, 3-11  
 GMDPRCDB.pls, 3-3  
 GMDPRCDS.pls, 3-3  
 GMDPRCFB.pls, 3-6  
 GMDPRCFS.pls, 3-6  
 GMDPRCHB.pls, 3-2  
 GMDPRCHS.pls, 3-2  
 GMDPROUB.pls, 1-8  
 GMDPROUS.pls, 1-8  
 GMDPRTSB.pls, 1-8  
 GMDPRTSS.pls, 1-8  
 GMDPVRFB.pls, 3-7  
 GMDPVRFs.pls, 3-7  
 GMDRVALB.pls, 3-5  
 GMDRVALS.pls, 3-5  
 GMDVFMDB.pls, 1-8  
 GMDVFMDS.pls, 1-8  
 GMDVFMHB.pls, 1-8  
 GMDVFMHS.pls, 1-8  
 GMDVOPRB.pls, 1-9

GMDVOPRS.pls, 1-9

## I

---

Input Parameters, 2-5  
 INSERT\_ACTIVITY, 4-47  
 INSERT\_OPERATION, 4-37  
 INSERT\_OPERATION\_ACTIVITY, 4-41  
 INSERT\_OPERATION\_RESOURCES, 4-46  
 INSERT\_ROUTING, 4-27  
 INSERT\_ROUTING\_STEPS, 4-31  
 INSERT\_STEP\_DEPENDENCY, 4-34

## O

---

Online User Interface (UI), 1-4  
 Oracle Forms, in calling function, 2-1  
 Oracle Messages, 1-6

## P

---

p\_api\_version, 3-15  
 p\_commit, 3-15  
 p\_formula\_insert\_tbl\_type, 2-5, 2-6  
 p\_formula\_update\_tbl\_type, 2-5  
 p\_init\_msg\_list, 3-15  
 p\_validation\_level, 3-15  
 package, 3-7, 3-8, 3-9, 3-10, 3-11, 3-12, 3-13, 3-14  
 packages, 3-2, 3-3, 3-5, 3-6  
 PL/SQL, 1-2, 2-5  
 procedures, 3-2, 3-3, 3-5, 3-6, 3-7, 3-8, 3-9, 3-10,  
 3-11, 3-12, 3-13, 3-14  
 PROCESS\_LOSS\_FOR\_UPDATE, 4-20

## Q

---

QC\_MIN\_MAX\_DATE, A-6

## R

---

RECIPE\_CUST\_EXISTS, 4-20  
 RECIPE\_DESCRIPTION, 4-20  
 RECIPE\_EXISTS, 4-20  
 RECIPE\_FOR\_UPDATE, 4-20  
 Recipe\_header\_tbl\_type, 4-2  
 RECIPE\_NAME, 4-20

RECIPE\_ORGN\_CODE, 4-20  
RECIPE\_ORGN\_OPERATIONS, 4-17  
RECIPE\_ORGN\_RESOURCES, 4-18  
RECIPE\_ROUTING\_STEPS, 4-18

## **S**

---

Standard Parameters, 3-15  
Stored procedures, 1-2  
success, 1-2  
support policy, 1-3  
SY\_INVALID\_UM\_CODE, A-6  
SY\_REQUIRED, A-6

## **U**

---

UNDELETE\_ROUTING, 4-28  
unexpected, 1-2  
unknown, 1-2  
UOM\_CONVERSION\_MESG, 4-23  
UPDATE\_ACTIVITY, 4-47  
UPDATE\_OPERATION, 4-38  
UPDATE\_OPERATION\_ACTIVITY, 4-41  
UPDATE\_OPERATION\_RESOURCES, 4-46  
UPDATE\_RECIPE\_CUSTOMERS, 4-14  
UPDATE\_RECIPE\_HEADER, 4-8  
UPDATE\_RECIPE\_PROCESS\_LOSS, 4-15  
UPDATE\_RECIPE\_VR, 4-16  
UPDATE\_ROUTING, 4-27  
UPDATE\_ROUTING\_STEPS, 4-31  
UPDATE\_STEP\_DEPENDENCY, 4-34

## **V**

---

Value-ID Conversion, 3-16

## **W**

---

wrapper function, 2-1, 2-5

## **X**

---

x\_msg\_count, 3-15  
x\_msg\_data, 3-15  
x\_return\_status, 3-15

