Oracle® Supply Chain Management APIs and Open Interfaces

Guide

Release 12

Part No. B40113-02

September 2007

ORACLE®

Oracle Supply Chain Management APIs and Open Interfaces Guide, Release 12

Part No. B40113-02

Primary Author: Debani Banerjee

Contents

3 Oracle ASCP and Oracle Global ATP Server Open Interfaces - Part 2

4 Oracle ASCP and Oracle Global ATP Server Open Interfaces - Part 3

7 Cost Management Open Interfaces

8 Engineering Open Interfaces

9 Oracle Flow Manufacturing Open Interfaces and APIs

10 Oracle Shop Floor Management Open Interfaces

11 Oracle Quality Open Interfaces and APIs

12 Oracle Work in Process Open Interfaces

13 Oracle Inventory Open Interfaces and APIs

14 Oracle Items Open Interfaces and APIs

15 Oracle Warehouse Management Open Interfaces and APIs

16 Receiving Open Interface

17 eAM Open Interfaces and APIs

21 Oracle Asset Tracking API

22 Oracle Service Contracts Public APIs

Index

Send Us Your Comments

Oracle Supply Chain Management APIs and Open Interfaces Guide, Release 12

Part No. B40113-02

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

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

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

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

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

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

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

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

Preface

Intended Audience

Welcome to Release 12 of the *Oracle Supply Chain Management APIs and Open Interfaces Guide*.

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

TTY Access to Oracle Support Services

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

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that

consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

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

Structure

Related Information Sources

Integration Repository

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

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

Do Not Use Database Tools to Modify Oracle Applications Data

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

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

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

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

1

Integrating Your Systems

This chapter covers the following topics:

- Overview
- Overview of Oracle Supply Chain Management APIs and Open Interfaces

Overview

This chapter provides an overview of Oracle Supply Chain Management integration tools and explains how to use these tools to integrate Oracle Supply Chain Management products with one another and with your existing non–Oracle systems.

Oracle Supply Chain Management integration tools are powerful, flexible tools that enable you to capture data from other Oracle applications or your own applications, define necessary format conversions, and direct data to your Oracle Supply Chain Management products. Topics covered here include:

Overview of Oracle Supply Chain Management APIs and Open Interfaces

Overview of Oracle Supply Chain Management APIs and Open Interfaces

Oracle Supply Chain Management products provide a number of open interfaces that enable you to link with non-Oracle applications, applications you build, applications on other computers, and even the applications of your suppliers and customers.

The purpose of this essay is to help you understand the general model Oracle Supply Chain Management products use for open application interfaces. Other essays in this chapter provide specific information on how to use each of the open interfaces. Additional functional information on these interfaces is available in each product's User's Guide. Additional technical information on these interfaces is available in each product's technical reference manual.

Basic Business Needs

Oracle Supply Chain Management product application programming interfaces (APIs) and open interfaces provide you with the features that you need to support the following basic business needs:

- Connect to data collection devices. This feature lets you collect material movement transactions such as receipts, issues, quality data, movements, completions, and shipments. This feature speeds data entry and improves transaction accuracy.
- Connect to other systems (such as finite scheduling packages, computer-aided design systems, custom, and legacy manufacturing systems) to create integrated enterprise-wide systems.
- Connect to external systems (such as the customer's purchasing system and the supplier's order entry system) to better integrate the supply chain through electronic commerce.
- Control processing of inbound data imported from outside Oracle applications.
- Validate imported data to ensure integrity of Oracle Supply Chain Management products.
- Review, update, and resubmit imported data that failed validation.
- Export data from Oracle Supply Chain Management products.

Oracle Supply Chain Management Interfaces

Open Interfaces Architectures

Oracle Supply Chain Management products use three different methods to import and export data:

- Interface Tables
- Interface Views (Business Views)
- Function Calls or Programmatic Interfaces (Processes)

Interface Tables

Interface tables, both inbound and outbound, normally require some validation through a concurrent program. These tables are fully documented in the essays that follow this chapter.

In several instances, interfaces do not require an intermediate validation step. You can write directly to the product's tables after you consult the product's technical reference manual.

Interface Views (Business Views)

Views simplify the data relationships for easier processing, whether for reporting or data export. Oracle Supply Chain Management and Distribution products have defined business views that identify certain areas of key business interest. You can access this data using your tool of choice. The MTL_ITEM_QUANTITIES_VIEW is an example of a key business view.

Product views are defined in the technical reference manuals. The view definitions also briefly describe how they are used. Many views, such as shortage reporting views in Oracle Work in Process, have been added specifically for easier reporting. Dynamic Views have also been added in Oracle Quality. Dynamic Views are views that are dynamically created and re-created as you create and modify collection plans in Oracle Quality.

Function Calls or Programmatic Interfaces (Processes)

Some open interfaces are more fundamental to the architecture of Oracle Supply Chain Management products. They are not interfaces as much as open integration.

For example, note flex-field validation by table/view. In this class of inbound interfaces, the addition of views automatically imports data into an existing function. This provides tight integration without adding a batch process to move data.

Another example is modifying open stored procedures. In the Oracle Bills of Material concurrent program AutoCreate Configuration, you can add business-specific logic to match configurations (check for duplicate configurations) by modifying the stored procedures provided for this purpose.

Summary: Beyond Published Interfaces

The Oracle Cooperative Applications Initiative references many third-party products that provide import and export capabilities and enable loose-to-tight integration with legacy systems, other supplier systems, and so on. Contact your Oracle consultant for more information about system integration.

Current Documentation For Open Interfaces

Here are the actual names of the tables, views, and modules:

Key	-
Data Flow Direction	*Inbound* means into Oracle Supply Chain Management; *Outbound* means out from Oracle Supply Chain Management
Iface Man	The interface is documented in detail in the *Oracle Supply Chain Management and Distribution Open Interfaces Manual*

TRM	The tables, views, or modules are described in the product's technical reference manual

Interface/API Name	Data Flow Direction	Table, View, Process, or Procedure	Iface Man	TRM	Table, View, Module Name, or Procedure Name
INV	**INV**	**INV**	**INV**	**INV**	**INV**
Transactions	Inbound	Table	Yes	Yes	MTL_TRANSACTIONS_INTERFACE MTL_SERIAL_NUMBERS_INTERFACE MTL_TRANSACTION_LOTS_INTERFACE
Demand Interface	Inbound	Table	Yes	Yes	MTL_DEMAND_INTERFACE
On-Hand Balances	Outbound	View		Yes	MTL_ITEM_QUANTITIES_VIEW
User-Defined Supply	Inbound	Table		Yes	MTL_USER_SUPPLY
User-Defined Demand	Inbound	Table		Yes	MTL_USER_DEMAND
Replenishment	Inbound	Table	Yes	Yes	MTL_REPLENISH_HEADERS_INT MTL_REPLENISH_LINES_INT

Item	Inbound	Table	Yes	Yes	MTL_SYSTEM_ITEMS_INTERFACE MTL_ITEMS_REVISIONS_INTERFACE
Customer Item	Inbound	Table	Yes	Yes	MTL_CI_INTERFACE
Customer_item Cross-References	Inbound	Table	Yes	Yes	MTL_CI_XREFS_INTERFACE
Material	Inbound	Table	Yes	Yes	MTL_TRANSACTIONS_INTERFACE MTL_SERIAL_NUMBERS_INTERFACE MTL_TRANSACTION_LOTS_INTERFACE
ENG/BOM	**ENG/BOM**	**ENG/BOM**	**ENG/BOM**	**ENG/BOM**	**ENG/BOM**
MFG Calendar	Outbound	View		Yes	BOM_CALENDAR_MONTHS_VIEW

Bill of Material	Inbound	Table	Yes	Yes	BOM_BILL_OF_MTLS_INTERFACE BOM_INVENTORY_COMPS_INTERFACE BOM_REF_DESGS_INTERFACE BOM_SUB_COMPS_INTERFACE MTL_ITEMS_REVISIONS_INTERFACE BOM_EXPORT_TAB BOM_SMALL_EXPL_TEMP
Routings	Inbound	Table	Yes	Yes	BOM_OP_ROUTINGS_INTERFACE BOM_OP_SEQUENCES_INTERFACE BOM_OP_RESOURCES_INTERFACE MTL_RTG_ITEM_REVS_INTERFACE

ECO	Inbound	Table	Yes	Yes	ENG_ENG_CHANGES_INTERFACE ENG_ECO_REVISIONS_INTERFACE ENG_REVISED_ITEMS_INTERFACE BOM_INVENTORY_COMPS_ INTERFACE BOM_REF_DESGS_INTERFACE BOM_SUB_COMPS_INTERFACE
eAM	**eAM**	**eAM**	**eAM**	**eAm**	**eAM**
eAM Item Open Interface	Inbound	Table	Yes	Yes	MTL_SYSTEM_ITEMS_INTERFACE MTL_ITEM_REVISIONS_INTERFACE MTL_ITEM_CATEGORIES_ INTERFACE MTL_INTERFACE_ERRORS
eAM Asset Number Open Interface	Inbound	Table	Yes	Yes	MTL_EAM_ASSET_NUM_ INTERFACE MTL_EAM_ATTR_VAL_INTERFACE MTL_EAM_ATTR_VAL_INTERFACE

eAM Asset Genealogy Open Interface	Inbound	Table	Yes	Yes	MTL_OBJECT_GENEALOGY_INTERFACE
eAm Work Order Open Interface	Inbound	Table	Yes	Yes	WIP_JOB_SCHEDULE_INTERFACE WIP_DISCRETE_JOBS
eAM Meter Reading Open Interface	Inbound	Table	Yes	Yes	EAM_METER_READING_INTERFACE
Oracle Cost Management (see *Oracle Bills of Material Technical Reference Manual*)	-	-	-	-	-
Item Cost Inquiry	Outbound	View		Yes	CST_INQUIRY_TYPES CSTFQVIC (View Item Cost Information)
MFG Cost Reporting	Outbound	View		Yes	CST_REPORT_TYPES CSTRFICR (Inventory Valuation Report)
MRP/SCP	**MRP/SCP**	**MRP/SCP**	**MRP/SCP**	**MRP/SCP**	**MRP/SCP**
Forecast Interface	Inbound	Table	Yes	Yes	MRP_FORECAST_INTERFACE

Forecast Entries API	Inbound	Process PL/SQL Table	Yes	Yes	T_FORECAST_INTERFACE T_FORECAST_DESIGNATOR
Master Schedule Interface	Inbound	Table	Yes	Yes	MRP_SCHEDULE_INTERFACE
Master Schedule Relief Interface	Inbound	Table		Yes	MRP_RELIEF_INTERFACE
Planner Workbench Interface	Outbound	Process		Yes	Stored Procedure MRPPL06, or WIP_JOB_SCHEDULE_INTERFACE PO_REQUISITIONS_INTERFACE PO_RESCHEDULE_INTERFACE
Projected Requirements Interface	Outbound	Table		Yes	MRP_RECOMMENDATIONS
Projected Supply Interface	Outbound	Table		Yes	MRP_GROSS_REQUIREMENTS

Sourcing Rule API	Outbound	Procedure	Yes	Yes	MRP_SOURCING_RULE_PUB.PROCESS_SOURCING_RULE MRP_SRC_ASSIGNMENT_PUB.PROCESS_ASSIGNMENT
PO	**PO**	**PO**	**PO**	**PO**	**PO**
Requisitions	Inbound	Table	Yes	Yes	PO_REQUISITIONS_INTERFACE PO_REQ_DIST_INTERFACE
Requisition Reschedule	Inbound	Table	Yes	Yes	PO_RESCHEDULE_INTERFACE
Purchasing Documents	Inbound	Table	Yes	Yes	PO_HEADERS_INTERFACE PO_LINES_INTERFACE PO_DISTRIBUTIONS_INTERFACE
Receiving	Inbound	Table	Yes	Yes	RCV_HEADERS_INTERFACE RCV_TRANSACTIONS_INTERFACE
QA	**QA**	**QA**	**QA**	**QA**	**QA**
Collection Import	Inbound	Table	Yes	Yes	QA_RESULTS_INTERFACE

Dynamic Collection Plan View	Outbound	View	Yes	Yes	Q_COLLECTION_PLAN_NAME_V
Dynamic Collection Import View	Inbound	View	Yes	Yes	Q_COLLECTION_PLAN_NAME_IV
WSM	**WSM**	**WSM**	**WSM**	**WSM**	**WSM**
Import Lot Jobs	Inbound	Table	Yes	Yes	WSM_LOT_JOB_INTERFACE
Lot Move Transactions	Inbound	Table	Yes	Yes	WSM_LOT_MOVE_TXN_INTERFACE
WIP Lot Transactions	Inbound	Table	Yes	Yes	WSM_SPLIT_MERGE_TXN_INTERFACE WSM_STARTING_JOBS_INTERFACE WSM_RESULTING_JOBS_INTERFACE
Inventory Lot Transactions	Inbound	Table	Yes	Yes	WSM_LOT_SPLIT_MERGES_INTERFACE WSM_STARTING_LOTS_INTERFACE WSM_RESULTING_LOTS_INTERFACE
WIP	**WIP**	**WIP**	**WIP**	**WIP**	**WIP**

Move Transaction	Inbound	Table	Yes	Yes	WIP_MOVE_TXN_INTERFACE CST_COMP_SNAP_INTERFACE (if organization uses average costing)
Resource Transaction	Inbound	Table	Yes	Yes	WIP_COST_TXN_INTERFACE
Work Order Interface	Inbound	Table	Yes	Yes	WIP_JOB_SCHEDULE_INTERFACE WIP_JOB_DTLS_INTERFACE
WMS	**WMS**	**WMS**	**WMS**	**WMS**	**WMS**
Compliance Label	Inbound	Procedure	Yes	Yes	PRINT_LABEL_MANUAL_WRAP PRINT_LABEL PRINT_LABEL_WRAP
Device Integration	Inbound	Procedure	Yes	Yes	WMS_DEVICE_INTEGRATION_PVT.DEVICE_REQUEST WMS_DEVICE_INTEGRATION_PUB.SYNC_DEVICE_REQUEST

Locator Maintenance	Inbound	Procedure	Yes	Yes	CREATE_LOCATOR UPDATE_LOCATOR CREATE_LOC_ITEM_TIE DELETE_LOCATOR

Container	Inbound	Procedure	Yes	Yes	GENERATE_LPN_CP GENERATE_LPN ASSOCIATE_LPN CREATE_LPN MODIFY_LPN MODIFY_LPN_WRAPPER PACKUNPACK_CONTAINER PACK_PREPACK_CONTAINER VALIDATE_UPDATE_WT_VOLUME PURGE_LPN EXPLODE_LPN TRANSFER_LPN_CONTENTS CONTAINER_REQUIRED_QTY GET_OUTERMOST_LPN GET_LPN_LIST PREPACK_LPN_CP PREPACK_LPN PRINT_CONTENT_REPORT LPN_PACK_COMPLETE
WMS Installation	Inbound	Procedure	Yes	Yes	CHECK_INSTALL

Inbound Open Interface Model

Oracle Supply Chain Management products provide both inbound and outbound interfaces. For inbound interfaces, where these products are the destination, Oracle provides interface tables, as well as supporting validation, processing, and maintenance programs. For outbound interfaces, where these products are the source, Oracle provides database views. The destination application should provide the validation, processing, and maintenance programs.

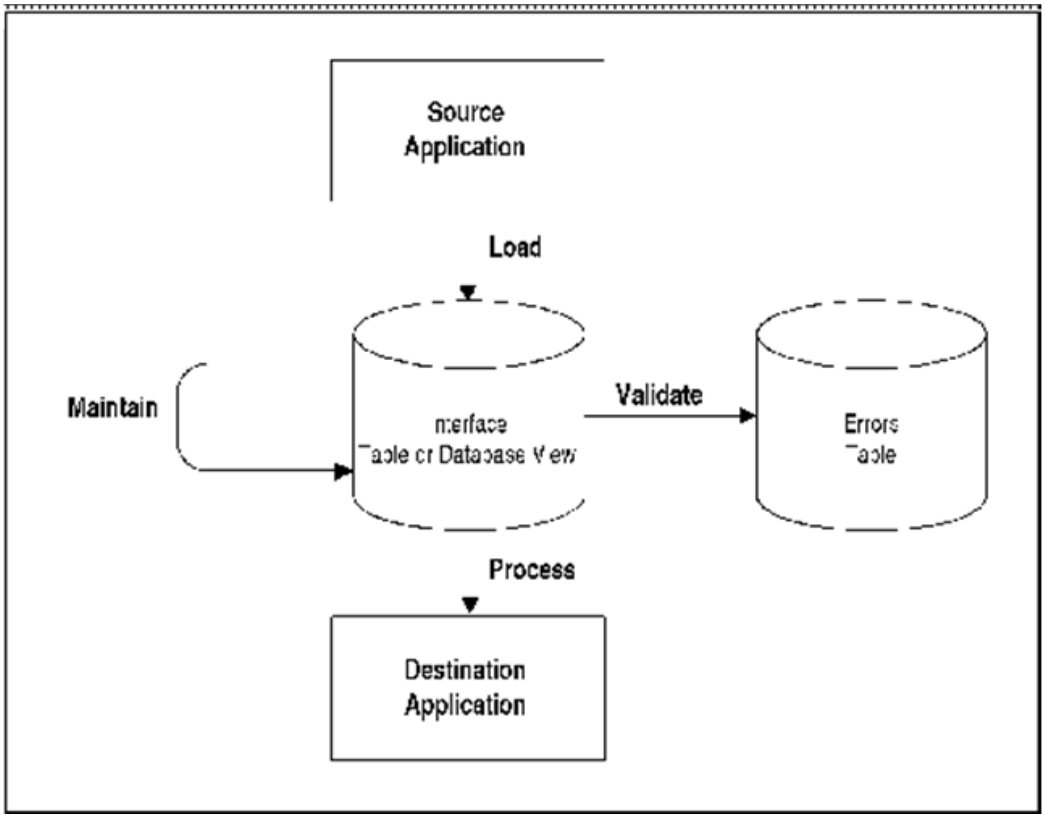
Discussion of Inbound Interfaces Only

This overview and the rest of the documents in this chapter discuss only inbound interfaces in detail. You can find information about the tables, views, and processes involved in outbound interfaces in the product's technical reference manual. Note that the technical reference manuals do not contain detailed, narrative discussions about the outbound interfaces.

Open Interface Diagram

The general model for open application interfaces is as follows:

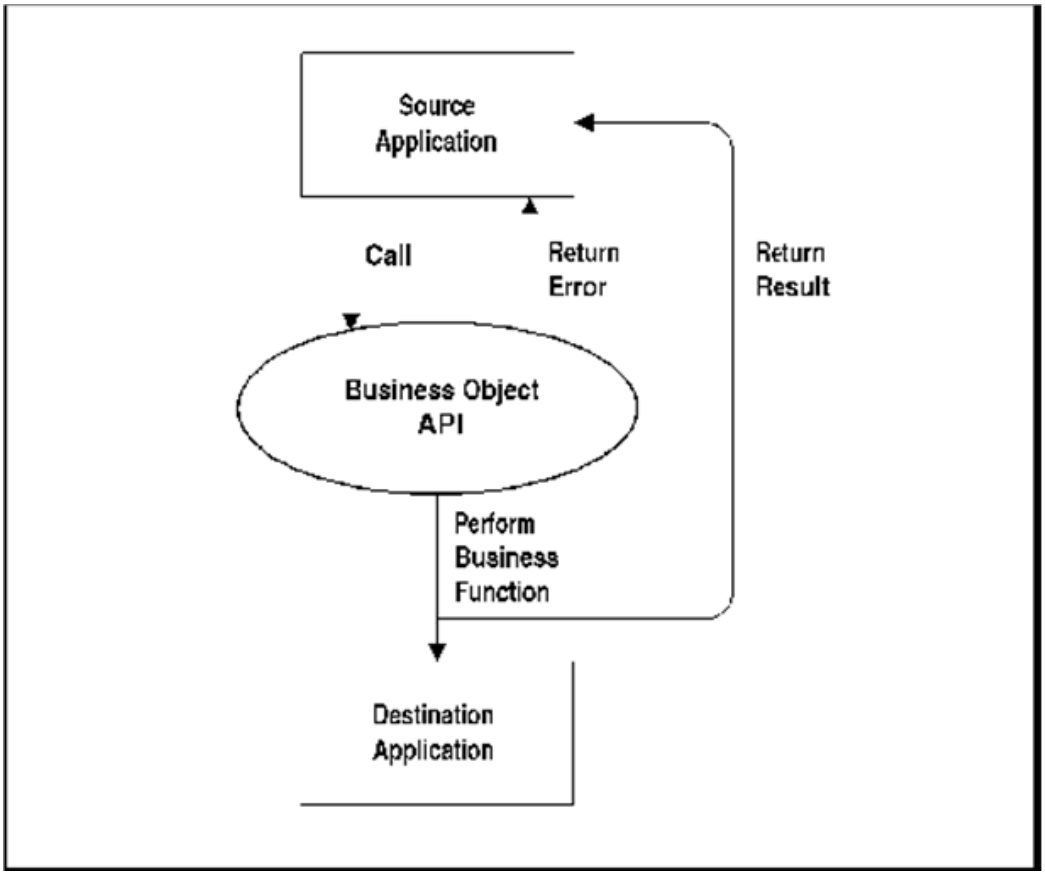
Open Interface Diagram



Open API Diagram

The model used by APIs such as the Service Request interfaces (Oracle Service) is as follows:

Open API Diagram



Components of an Open Interface

There are a number of components that are generally common to all open interfaces. This section discusses these components. However, all open interfaces do not include every component, and in some cases, you can implement the component slightly differently than described here.

Source Application

You obtain data from a source application to pass on to a destination application for further processing and storage. Typically, the data completes processing in the source application before it is passed.

Oracle Supply Chain Management products are the source for outbound interfaces. For example, Oracle Inventory is the source for the On-Hand Balances Interface. This interface is used to export on-hand balances from Oracle Inventory for use by other planning and distribution destination applications.

Destination Application You send data to a destination application so that the application can perform further processing and storage.

Oracle Supply Chain Management products are the destinations for inbound interfaces. For example, Oracle Purchasing is the destination for receiving transactions imported using the Receiving Open Interface. Oracle Purchasing updates purchase orders for each receiving transaction and creates and stores the receiving transaction history.

Interface Table

For inbound interfaces, the interface table is the intermediary table where the data from the source application temporarily resides until it is validated and processed into an Oracle Supply Chain Management product. The various types of interface columns, with examples from the Oracle Work in Process Move Transaction Interface, are listed here:

Identifier Columns Identifier columns uniquely identify rows in the interface table and provide foreign key reference to both the source and destination applications. For example, typical identifier columns for a move transaction would identify:

- The source application, such as the bar code device identifier
- The row's unique identifier in the source application, such as the job name
- The destination application's unique identifier, such as the Work in Process entity ID.

Control Columns Control columns track the status of each row in the interface table as it is inserted, validated, errored, processed, and ultimately deleted. Additional control columns identify who last updated the row and the last update date.

Data Columns Data columns store the specific attributes that the source application is sending to the Oracle Supply Chain Management product. For example, transaction quantity is one attribute of a move transaction.

Required Columns Required columns store the minimum information needed by the Oracle Supply Chain Management product to successfully process the interface row. For example, an organization code is required for all move transactions.

Some columns are conditionally required based on the specifics of the interface. For example, repetitive move transactions require production line information, whereas discrete move transactions do not.

Derived Columns The destination product creates the derived columns from information in the required columns. For example, on a move transaction, the primary unit of measure is derived from the assembly that you are moving.

Optional Columns Oracle Supply Chain Management products do not necessarily require optional columns but you can use them for additional value-added functionality. For example, for move transactions, the reason code is not required, but you can use the code to collect additional transaction information.

Errors Table

For inbound interfaces, the errors table stores all errors found by the validation and processing functions. In some cases, the errors table is a child of the interface table. This

enables each row in the interface table to have many errors, so that you can manage multiple errors at once. In other cases, the errors are stored in a column within the interface table. In this case, you must fix each error independently.

For example, in the Oracle Work in Process Open Resource Transaction Interface, the validation program inserts an error into an errors table when resource transaction records fail validation because of a missing piece of required data, such as the resource transaction quantity. In contrast, the Order Import process in Oracle Order Management/Shipping inserts errors into a single errors column in the interface table when rows fail validation.

Database View

Database views are database objects that make data from the Oracle Supply Chain Management source products available for selection and use by destination applications.

Oracle Supply Chain Management products provide predefined views of key data that is likely to be used by destination applications. In addition to the predefined views that these products use, Oracle Quality also provides non-predefined, dynamic views. These views join related tables within source products so that the destination application can select the data.

For example, Oracle Cost Management provides work in process valuation and transaction distribution database views for use by other cost reporting destination products.

Load Function

For inbound interfaces, the load function is the set of programs that selects and accumulates data from the source application and inserts it into Oracle Supply Chain Management interface tables. The programming languages and tools used in the load function are highly dependent on the hardware and system software of the source application.

For example, if you are passing data between an Oracle-based source application and an Oracle Supply Chain Management product, you would likely use a tool such as Pro*C or PL/SQL because these tools work in both environments. If you are bringing data from a non Oracle-based application into a product's interface table, you would likely use a procedural language available on the source application to select the data and convert it into an ASCII file. Then you could use SQL*Loader to insert that file into the destination product's interface table.

For outbound interfaces, the load function is the SQL that creates the database view. For example, the Item Cost Interface in Oracle Cost Management uses SQL to create several database views of the item cost information for use by other budgeting and cost analysis destination applications.

Validate Function

The validate function is the set of programs that Oracle Supply Chain Management destination products use to ensure the integrity of inbound data. In the source

application, you can typically validate data upon entry using techniques such as forms triggers, not null columns, data types, and so on. However, because Oracle Supply Chain Management products are not the source of this data, validation programs ensure data integrity.

In addition, the validate function can derive additional columns based on the required columns and foreign key relationships with other data elsewhere in the Oracle Supply Chain Management destination application.

The validation programs check the interface table for rows requiring validation, then validate and update each row indicating either validation complete or errors found. If errors are found, validation programs must write errors to the destination application's errors table or to the interface table's error column.

For example, the move transaction validation program performs several validation tasks within the Oracle Work in Process Open Move Transaction Interface. These tasks include:

- checking the accuracy of specific columns such as the job or schedule name
- checking the completeness of each row such as the transaction unit of measure and transaction quantity
- checking the relationship between columns in the same row such as the from and to operation sequence numbers

2

Oracle ASCP and Oracle Global ATP Server Open Interfaces - Part 1

This chapter covers the following topics:

- ODS Load API Features
- Functional Overview
- Setting Up the ODS Load API
- Parameter Descriptions
- MSC_ST_ASSIGNMENT_SETS
- MSC_ST_ATP_RULES
- MSC_ST_BILL_OF_RESOURCES
- MSC_ST_BIS_BUSINESS_PLANS
- MSC_ST_BIS_PERIODS
- MSC_ST_BIS_PFMC_MEASURES
- MSC_ST_BIS_TARGETS
- MSC_ST_BIS_TARGET_LEVELS
- MSC_ST_BOMS
- MSC_ST_BOM_COMPONENTS
- MSC_ST_BOR_REQUIREMENTS
- MSC_ST_CALENDAR_DATES
- MSC_ST_CALENDAR_SHIFTS
- MSC_ST_CAL_WEEK_START_DATES
- MSC_ST_CAL_YEAR_START_DATES
- MSC_ST_CATEGORY_SETS

- MSC_ST_COMPONENT_SUBSTITUTES
- MSC_ST_DEMANDS
- MSC_ST_DEMAND_CLASSES
- MSC_ST_DEPARTMENT_RESOURCES
- MSC_ST_DESIGNATORS
- MSC_ST_INTERORG_SHIP_METHODS

ODS Load API Features

The ODS API consists of the following entities (staging tables):

- Inventory Items information

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update information for existing items.

  The following staging tables are used:

  - MSC_ST_SYSTEM_ITEMS
  - MSC_ST_SAFETY_STOCKS

- Sourcing Rules

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing sourcing information.

  The following tables are used:

  - MSC_ST_ASSIGNMENT_SETS
  - MSC_ST_SOURCING_RULES
  - MSC_ST_SR_ASSIGNMENTS
  - MSC_ST_SR_RECEIPT_ORG
  - MSC_ST_SR_SOURCE_ORG

    In complete refresh mode, you can renew all entries using the table, MSC_ST_INTERORG_SHIP_METHODS

    In both complete and refresh mode, you can update the sourcing history information using the table, MSC_ST_SOURCING_HISTORY

- ATP Rules

  In complete refresh mode, you can renew all entries using the table,

MSC_ST_ATP_RULES

- Bill of Resources

  In complete refresh mode, you can renew all entries. The following tables are used:

  - MSC_ST_BILL_OF_RESOURCES
  - MSC_ST_BOR_REQUIREMENTS

- BOMs/Routings

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing BOM/Routing data.

  The following tables are used:

  - MSC_ST_PROCESS_EFFECTIVITY
  - MSC_ST_BOMS
  - MSC_ST_BOM_COMPONENTS
  - MSC_ST_COMPONENT_SUBSTITUTES
  - MSC_ST_ROUTINGS
  - MSC_ST_ROUTING_OPERATIONS
  - MSC_ST_OPERATION_RESOURCE_SEQS
  - MSC_ST_OPERATION_RESOURCES
  - MSC_ST_OPERATION_COMPONENTS

- Calendar system

  In complete refresh mode, you can renew all entries. The following tables are used:

  - MSC_ST_CALENDAR_DATES
  - MSC_ST_CAL_YEAR_START_DATES
  - MSC_ST_CAL_WEEK_START_DATES
  - MSC_ST_PERIOD_START_DATES
  - MSC_ST_CALENDAR_SHIFTS
  - MSC_ST_SHIFT_DATES

- MSC_ST_SHIFT_TIMES
- MSC_ST_SHIFT_EXCEPTIONS
- MSC_ST_SIMULATION_SETS
- MSC_ST_RESOURCE_SHIFTS

- Categories

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing categories information.

  The following tables are used:

  - MSC_ST_CATEGORY_SETS
  - MSC_ST_ITEM_CATEGORIES

- MDS/MPS designators

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing designators information.

  These operations use the table, MSC_ST_DESIGNATORS

- Demands and Sales Orders

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing demands and sales orders information.

  The following tables are used:

  - MSC_ST_DEMANDS
  - MSC_ST_RESERVATIONS
  - MSC_ST_SALES_ORDERS

- Supplies

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing supplies information.

  These operations use the table, MSC_ST_SUPPLIES

- Resources

  In complete refresh mode, you can renew all entries. The following tables are used:

  - MSC_ST_RESOURCE_GROUPS

- MSC_ST_DEPARTMENT_RESOURCES

  In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing resources information.

  The following tables are used:

- MSC_ST_NET_RESOURCE_AVAIL
- MSC_ST_RESOURCE_REQUIREMENTS
- MSC_ST_RESOURCE_CHANGES

- Approved Suppliers Information

  In complete refresh mode, you can renew all entries. The following tables are used:

  - MSC_ST_ITEM_SUPPLIERS
  - MSC_ST_SUPPLIER_FLEX_FENCES

    In complete refresh mode, you can renew all entries. In incremental refresh mode, you can create new entries, and update existing supplier capacities information.

    These operations use the table, MSC_ST_SUPPLIER_CAPACITIES

- Trading Partners Information

  In complete refresh mode, you can renew all entries. The following tables are used:

  - MSC_ST_TRADING_PARTNERS
  - MSC_ST_TRADING_PARTNER_SITES
  - MSC_ST_LOCATION_ASSOCIATIONS
  - MSC_ST_PARAMETERS
  - MSC_ST_SUB_INVENTORIES
  - MSC_ST_PARTNER_CONTACTS

- Planner Information

  In complete refresh mode, you can renew all entries using the table, MSC_ST_PLANNERS

- Units Of Measure

  In complete refresh mode, you can renew all entries. The following tables are used:

- MSC_ST_UNITS_OF_MEASURE
- MSC_ST_UOM_CONVERSIONS
- MSC_ST_UOM_CLASS_CONVERSIONS

- Unit Number, Projects, and Tasks Information

  In complete refresh mode, you can renew all entries. The following tables are used:

  - MSC_ST_UNIT_NUMBERS
  - MSC_ST_PROJECTS
  - MSC_ST_PROJECT_TASKS

- BIS Objects

  In complete refresh mode, you can renew all entries.

  - MSC_ST_BIS_BUSINESS_PLANS
  - MSC_ST_BIS_PERIODS
  - MSC_ST_BIS_PFMC_MEASURES
  - MSC_ST_BIS_TARGET_LEVELS
  - MSC_ST_BIS_TARGETS

- Demand Classes

  In complete refresh mode, you can renew all entries using the table, MSC_ST_DEMAND_CLASSES

See Also

The *Oracle ASCP and Oracle Global ATP Server Technical Reference Manual*.

Functional Overview

The ODS Load API provides a public procedure, Launch_Monitor, for loading the data into the ODS tables.

The Launch_Monitor procedure performs the following major processes:

- Generate new local ID for the global attributes such as item, category set, vendor, vendor site, customer, and customer site.

- Launch the ODS Load Workers to perform Create, Update, and Delete Operation for each entity in ODS.
- Recalculate the sourcing history based on the latest sourcing information and the data from the transaction systems.
- Recalculate the net resource availability based on the calendars, shifts, and department resources information.
- Purge the data in the staging tables.

Setting Up the ODS Load API

The ODS Load API is a stored PL/SQL function. You need to define certain data before you create or update ODS data. Before using the API, set up and/or activate the following parameters:

- Instance Code
- Number of Workers
- Recalculate Net Resource Availability (Yes/No)
- Recalculate Sourcing History (Yes/No)

Parameter Descriptions

The following charts describe all staging tables used by the ODS Load API. All of the inbound and outbound parameters are listed for these table. Additional information on these parameters follows each chart.

MSC_ST_ASSIGNMENT_SETS

The staging table used by the collection program to valid and process data for table MSC_ASSIGNMENT_SETS.

Parameter	Usage	Type	Required	Derived	Optional
SR_ASSIGNMENT_SET_ID	**IN**	**NUMBER**	**x**		
ASSIGNMENT_SET_NAME	IN	VARCHAR2(34)	x		
DESCRIPTION	IN	VARCHAR2(80)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE	x		

Parameter	Usage	Type	Required	Derived	Optional
SR_ASSIGNMENT_SET_ID	**IN**	**NUMBER**	**x**		
SR_INSTANCE_ID	IN	NUMBER			x
REFRESH_ID	IN	NUMBER			x

SR_ASSIGNMENT_SET_ID

Assignment set identifier from source application instance

ASSIGNMENT_SET_NAME

Assignment set name

DESCRIPTION

Description

DELETED_FLAG

Flag to indicate whether the row is no longer valid. SYS_YES means the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_ATP_RULES

The staging table used by the collection program to validate and process data for table MSC_ATP_RULES.

Parameter	Usage	Type	Required	Derived	Optional
RULE_ID	IN	NUMBER	x		
SR_INSTANCE_ID	IN	NUMBER	x		
RULE_NAME	IN	VARCHAR2(80)	x		
DESCRIPTION	IN	VARCHAR2(240)			x
ACCUMULATE_AVAILABLE_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
BACKWARD_CONSUMPTION_FLAG	IN	NUMBER	x		
FORWARD_CONSUMPTION_FLAG	IN	NUMBER	x		
PAST_DUE_DEMAND_CUTOFF_FENCE	IN	NUMBER			x
PAST_DUE_SUPPLY_CUTOFF_FENCE	IN	NUMBER			x
INFINITE_SUPPLY_FENCE_CODE	IN	NUMBER	x		
INFINITE_SUPPLY_TIME_FENCE	IN	NUMBER			x
ACCEPTABLE_EARLY_FENCE	IN	NUMBER			x
ACCEPTABLE_LATE_FENCE	IN	NUMBER			x
DEFAULT_ATP_SOURCES	IN	NUMBER			x
INCLUDE_SALES_ORDERS	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
INCLUDE_DISCRETE_WIP_DEMAND	IN	NUMBER	x		
INCLUDE_REP_WIP_DEMAND	IN	NUMBER	x		
INCLUDE_NONSTD_WIP_DEMAND	IN	NUMBER	x		
INCLUDE_DISCRETE_MPS	IN	NUMBER	x		
INCLUDE_USER_DEFINED_DEMAND	IN	NUMBER	x		
INCLUDE_PURCHASE_ORDERS	IN	NUMBER	x		
INCLUDE_DISCRETE_WIP_RECEIPTS	IN		x		
INCLUDE_REP_WIP_RECEIPTS	IN	NUMBER	x		
INCLUDE_NONSTD_WIP_RECEIPTS	IN	NUMBER	x		
INCLUDE_INTERORG_TRANSFERS	IN	NUMBER	x		
INCLUDE_ONHAND_AVAILABLE	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
INCLUDE_USER_DEFINED_SUPPLY	IN	NUMBER	x		
ACCUMULATION_WINDOW	IN	NUMBER			x
INCLUDE_REP_MPS	IN	NUMBER	x		
INCLUDE_INTERNAL_REQS	IN	NUMBER			x
INCLUDE_SUPPLIER_REQS	IN	NUMBER			x
INCLUDE_INTERNAL_ORDERS	IN	NUMBER			x
INCLUDE_FLOW_SCHEDULE_DEMAND	IN	NUMBER			x
USER_ATP_SUPPLY_TABLE_NAME	IN	VARCHAR2(30)			x
USER_ATP_DEMAND_TABLE_NAME	IN	VARCHAR2(30)			x
MPS_DESIGNATOR	IN	VARCHAR2(10)			x
LAST_UPDATE_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER		x	
DEMAND_CLASS_ATP_FLAG	IN	NUMBER	x		
INCLUDE_FLOW_SCHEDULE_RECEIPTS	IN	NUMBER	x		

RULE_ID

ATP rule identifier

SR_INSTANCE_ID

Instance id

RULE_NAME

Name of ATP rule

DESCRIPTION

Description for ATP rule

ACCUMULATE_AVAILABLE_FLAG

Flag for ATP computation to accumulate quantity availability

BACKWARD_CONSUMPTION_FLAG

Flag for ATP computation to backwardly consume shortage

FORWARD_CONSUMPTION_FLAG

Flag for ATP computation to forwardly consume shortage

PAST_DUE_DEMAND_CUTOFF_FENCE

Demand before the specified number of days are not to be considered in ATP computation

PAST_DUE_SUPPLY_CUTOFF_FENCE

Supplies before the specified number of days are not to be considered in ATP computation

INFINITE_SUPPLY_FENCE_CODE

Source code for infinite supply time fence

INFINITE_SUPPLY_TIME_FENCE

Infinite supply time fence days only when user-defined is specified in the time fence code

ACCEPTABLE_EARLY_FENCE

Acceptable early fence

ACCEPTABLE_LATE_FENCE

Acceptable late fence

DEFAULT_ATP_SOURCES

Indicate which subinventories to use for on-hand quantities

INCLUDE_SALES_ORDERS

Yes/No flag for ATP computation to include demand from sales orders

INCLUDE_DISCRETE_WIP_DEMAND

Yes/No flag for ATP computation to include demand from WIP discrete jobs

INCLUDE_REP_WIP_DEMAND

Yes/No flag for ATP computation to include demand from WIP repetitive discrete jobs

INCLUDE_NONSTD_WIP_DEMAND

Yes/No flag for ATP computation to include demand from WIP non-standard jobs'

INCLUDE_DISCRETE_MPS

Yes/No flag for ATP computation to include supply from discrete MPS schedule

INCLUDE_USER_DEFINED_DEMAND

Yes/No flag for ATP computation to include user defined demand

INCLUDE_PURCHASE_ORDERS

Yes/No flag for ATP computation to include supply from purchase orders

INCLUDE_DISCRETE_WIP_RECEIPTS

Yes/No flag for ATP computation to include supply from WIP discrete jobs

INCLUDE_REP_WIP_RECEIPTS

Yes/No flag for ATP computation to include supply from WIP repetitive schedule jobs

INCLUDE_NONSTD_WIP_RECEIPTS

Yes/No flag for ATP computation to include supply from WIP non-standard jobs

INCLUDE_INTERORG_TRA SFERS

Yes/No flag for ATP computation to include supply from inter-organization transfers

INCLUDE_ONHAND_AVAILABLE

Yes/No flag for ATP computation to include supply from on-hand inventory

INCLUDE_USER_DEFINED_SUPPLY

Yes/No flag for ATP computation to include supply from user defined source

ACCUMULATION_WINDOW

Maximum number of days that available supply should be accumulated

INCLUDE_REP_MPS

Yes/No flag for ATP computation to include supply from repetitive MPS schedules

INCLUDE_INTERNAL_REQS

Yes/No flag for ATP computation include from internal requisitions

INCLUDE_SUPPLIER_REQS

Yes/No flag for ATP computation include from internal orders

INCLUDE_INTERNAL_ORDERS

Yes/No flag for ATP computation to include demand from internal orders

INCLUDE_FLOW_SCHEDULE_DEMAND

Yes/No flag for ATP computation to include demand from flow schedule

USER_ATP_SUPPLY_TABLE_NAME

Not currently used

USER_ATP_DEMAND_TABLE_NAME

Not currently used

MPS_DESIGNATOR

Not currently used

LAST_UPDATE_DATE

Standard Who Column

LAST_UPDATED_BY

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

REFRESH_ID

Refresh identifier

DEMAND_CLASS_ATP_FLAG

Yes/No flag for ATP computation to consider Demand Class when selecting supply and demand

INCLUDE_FLOW_SCHEDULE_RECEIPTS

Yes/No flag for ATP computation to include supply from repetitive MPS schedules

MSC_ST_BILL_OF_RESOURCES

The staging table used by the collection program to validate and process data for table MSC_BILL_OF_RESOURCES.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
BILL_OF_RESOURCES	IN	VARCHAR2(10)	x		
DESCRIPTION	IN	VARCHAR2(50)			x
DISABLE_DATE	IN	DATE			x
ROLLUP_START_DATE	IN	DATE			x
ROLLUP_COMPLETION_DATE	IN	DATE			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

ORGANIZATION_ID

Organization identifier

BILL_OF_RESOURCES

Source application bill of resource identifier

DESCRIPTION

Bill of resource description

DISABLE_DATE

Bill of resource disable date

ROLLUP_START_DATE

Bill of resources load start date

ROLLUP_COMPLETION_DATE

Bill of resources load completion date

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID NULL

Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_BIS_BUSINESS_PLANS

The staging table used by the collection program to validate and process data for table MSC_BIS_BUSINESS_PLANS.

Parameter	Usage	Type	Required	Derived	Optional
BUSINESS_PLAN_ID	IN	NUMBER	x		
SHORT_NAME	IN	VARCHAR2(30)	x		
NAME	IN	VARCHAR2(80)	x		
DESCRIPTION	IN	VARCHAR2(240)			x
VERSION_NO	IN	NUMBER	x		
CURRENT_PLAN_FLAG	IN	VARCHAR2(1)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE			x
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

BUSINESS_PLAN_ID

Business plan identifier

SHORT_NAME

Business plan short name

NAME

Business plan name

DESCRIPTION

Describe the business plan

VERSION_NO

Version number

CURRENT_PLAN_FLAG

Yes/No flag indicating whether the business plan is current

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_BIS_PERIODS

The staging table used by the collection program to validate and process data for table MSC_BIS_PERIODS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
PERIOD_SET_NAME	IN	VARCHAR2(15)	x		
PERIOD_NAME	IN	VARCHAR2(15)	x		
START_DATE	IN	DATE	x		
END_DATE	IN	DATE	x		
PERIOD_TYPE	IN	VARCHAR2(15)	x		
PERIOD_YEAR	IN	NUMBER(15)	x		
PERIOD_NUM	IN	NUMBER(15)	x		
QUARTER_NUM	IN	NUMBER(15)	x		
ENTERED_PERIOD_NAME	IN	VARCHAR2(15)	x		
ADJUSTMENT_PERIOD_FLAG	IN	VARCHAR2(1)	x		
DESCRIPTION	IN	VARCHAR2(240)			x

Parameter	Usage	Type	Required	Derived	Optional
CONTEXT	IN	VARCHAR2(150)			x
YEAR_START_DATE	IN	DATE			x
QUARTER_START_DATE	IN	DATE			x
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

ORGANIZATION_ID

Organization identifier

PERIOD_SET_NAME

Accounting calendar name

PERIOD_NAME

Accounting calendar name

START_DATE

Date on which accounting period begins

END_DATE

Date on which accounting period ends

PERIOD_TYPE

Accounting period type

PERIOD_YEAR

Accounting period year

PERIOD_NUM

Accounting period number

QUARTER_NUM

Accounting period number

ENTERED_PERIOD_NAME

User entered accounting period name

ADJUSTMENT_PERIOD_FLAG

Calendar period adjustment status

DESCRIPTION

Accounting period description

CONTEXT

Descriptive flexfield segment

YEAR_START_DATE

Date on which the year containing this accounting period starts

QUARTER_START_DATE

Date on which the quarter containing this accounting period starts

LAST_UPDATE_DATE

Standard Who Column

LAST_UPDATED_BY

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_BIS_PFMC_MEASURES

The staging table used by the collection program to validate and process data for table MSC_BIS_PFMC_MEASURES.

Parameter	Usage	Type	Required	Derived	Optional
MEASURE_ID	IN	NUMBER	x		
MEASURE_SHORT_NAME	IN	VARCHAR2(30)	x		
MEASURE_NAME	IN	VARCHAR2(80)	x		
DESCRIPTION	IN	VARCHAR2(240)			x
ORG_DIMENSION_ID	IN	NUMBER			x
TIME_DIMENSION_ID	IN	NUMBER			x
DIMENSION1_ID	IN	NUMBER			x
DIMENSION2_ID	IN	NUMBER			x
DIMENSION3_ID	IN	NUMBER			x
DIMENSION4_ID	IN	NUMBER			x
DIMENSION5_ID	IN	NUMBER			x
UNIT_OF_MEASURE_CLASS	IN	VARCHAR2(10)			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

MEASURE_ID

Measure identifier

MEASURE_SHORT_NAME

Source application instance identifier

MEASURE_NAME

Measure short name

DESCRIPTION

Describe the performance measure

ORG_DIMENSION_ID

Organization dimension identifier

TIME_DIMENSION_ID

Time dimension identifier

DIMENSION1_ID

First dimension identifier

DIMENSION2_ID

Second dimension identifier

DIMENSION3_ID

Third dimension identifier

DIMENSION4_ID

Forth dimension identifier

DIMENSION5_ID

Fifth dimension identifier

UNIT_OF_MEASURE_CLASS

Unit of measure class

DELETED_FLAG

Yes/No flag indicates whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_BIS_TARGETS

The staging table used by the collection program to validate and process data for table MSC_BIS_TARGETS.

Parameter	Usage	Type	Required	Derived	Optional
TARGET_ID	IN	NUMBER	x		
TARGET_LEVEL_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
BUSINESS_PLAN_ID	IN	NUMBER	x		
ORG_LEVEL_VALUE_ID	IN	VARCHAR2(80)	x		
TIME_LEVEL_VALUE_ID	IN	VARCHAR2(80)			x
DIM1_LEVEL_VALUE_ID	IN	VARCHAR2(80)			x
DIM2_LEVEL_VALUE_ID	IN	VARCHAR2(80)			x
DIM3_LEVEL_VALUE_ID	IN	VARCHAR2(80)			x
DIM4_LEVEL_VALUE_ID	IN	VARCHAR2(80)			x
DIM5_LEVEL_VALUE_ID	IN	VARCHAR2(80)			x
TARGET	IN	NUMBER			x
RANGE1_LOW	IN	NUMBER			x
RANGE1_HIGH	IN	NUMBER			x
RANGE2_LOW	IN	NUMBER			x
RANGE2_HIGH	IN	NUMBER			x
RANGE3_LOW	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
RANGE3_HIGH	IN	NUMBER			x
NOTIFY_RESP1_ID	IN	NUMBER			x
NOTIFY_RESP1_SHORT_NAME	IN	VARCHAR2(100)			x
NOTIFY_RESP2_ID	IN	NUMBER			x
NOTIFY_RESP2_SHORT_NAME	IN	VARCHAR2(100)			x
NOTIFY_RESP3_ID	IN	NUMBER			x
NOTIFY_RESP3_SHORT_NAME	IN	VARCHAR2(100)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

TARGET_ID

Target identifier

TARGET_LEVEL_ID

Target level identifier

BUSINESS_PLAN_ID

Business plan identifier

ORG_LEVEL_VALUE_ID

Org level value identifier

TIME_LEVEL_VALUE_ID

Time level value identifier

DIM1_LEVEL_VALUE_ID

First dimension level value identifier

DIM2_LEVEL_VALUE_ID

Second dimension level value identifier

DIM3_LEVEL_VALUE_ID

Third dimension level value identifier

DIM4_LEVEL_VALUE_ID

Forth dimension level value identifier

DIM5_LEVEL_VALUE_ID

Fifth dimension level value identifier

TARGET

Target number

RANGE1_LOW

Low number of the first range

RANGE1_HIGH

High number of the first range

RANGE2_LOW

Low number of the second range

RANGE2_HIGH

High number of the second range

RANGE3_LOW

Low number of the third range

RANGE3_HIGH

High number of the third range

NOTIFY_RESP1_ID

First notify identifier

NOTIFY_RESP1_SHORT_NAME

Short name of the first notify

NOTIFY_RESP2_ID

Second notify identifier

NOTIFY_RESP2_SHORT_NAME

Short name of the second notify

NOTIFY_RESP3_ID

Third notify identifier

NOTIFY_RESP3_SHORT_NAME

Short name of the third notify

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_BIS_TARGET_LEVELS

The staging table used by the collection program to validate and process data for table MSC_BIS_TARGET_LEVELS.

Parameter	Usage	Type	Required	Derived	Optional
TARGET_LEVEL_ID	IN	NUMBER	x		
TARGET_LEVEL_SHORT_NAME	IN	VARCHAR2(30)	x		
TARGET_LEVEL_NAME	IN	VARCHAR2(80)	x		
DESCRIPTION	IN	VARCHAR2(240)	x		
MEASURE_ID	IN	NUMBER	x		
ORG_LEVEL_ID	IN	NUMBER	x		
TIME_LEVEL_ID	IN	NUMBER	x		
DIMENSION1_LEVEL_ID	IN	NUMBER			x
DIMENSION2_LEVEL_ID	IN	NUMBER			x
DIMENSION3_LEVEL_ID	IN	NUMBER			x
DIMENSION4_LEVEL_ID	IN	NUMBER			x
DIMENSION5_LEVEL_ID	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
WORKFLOW_ITEM_TYPE	IN	VARCHAR2(8)			x
WORKFLOW_PROCESS_SHORT_NAME	IN	VARCHAR2(30)			x
DEFAULT_NOTIFY_RESP_ID	IN	NUMBER			x
DEFAULT_NOTIFY_RESP_SHORT_NAME	IN	VARCHAR2(100)			x
COMPUTING_FUNCTION_ID	IN	NUMBER			x
REPORT_FUNCTION_ID	IN	NUMBER			x
UNIT_OF_MEASURE	IN	VARCHAR2(25)			x
SYSTEM_FLAG	IN	VARCHAR2(1)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

TARGET_LEVEL_ID

Target level identifier

TARGET_LEVEL_SHORT_NAME

Short name identifying the target level

TARGET_LEVEL_NAME

Target level name

DESCRIPTION

Describe the target level

MEASURE_ID

Performance measure identifier

ORG_LEVEL_ID

Organization level identifier

TIME_LEVEL_ID

Time level identifier

DIMENSION1_LEVEL_ID

First dimension level identifier

DIMENSION2_LEVEL_ID

Second dimension level identifier

DIMENSION3_LEVEL_ID

Third dimension level identifier

DIMENSION4_LEVEL_ID

Forth dimension level identifier

DIMENSION5_LEVEL_ID

Fifth dimension level identifier

WORKFLOW_ITEM_TYPE

Workflow item type

WORKFLOW_PROCESS_SHORT_NAME

Workflow process short name

DEFAULT_NOTIFY_RESP_ID

Default notify identifier

DEFAULT_NOTIFY_RESP_SHORT_NAME

Name of the default notify

COMPUTING_FUNCTION_ID

Computing function identifier

REPORT_FUNCTION_ID

Report function identifier

UNIT_OF_MEASURE

Unit of measure

SYSTEM_FLAG

System flag

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_BOMS

The staging table used by the collection program to valid and process data for table MSC_BOMS.

Parameter	Usage	Type	Required	Derived	Optional
BILL_SEQUENCE_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
ASSEMBLY_ITEM_ID	IN	NUMBER	x		
ASSEMBLY_TYPE	IN	NUMBER	x		
ALTERNATE_BOM_DESIGNATOR	IN	VARCHAR2(10)			x
SPECIFIC_ASSEMBLY_COMMENT	IN	VARCHAR2(240)			x
PENDING_FROM_ECN	IN	VARCHAR2(10)			x
COMMON_BILL_SEQUENCE_ID	IN	NUMBER			x
SCALING_TYPE	IN	NUMBER			x
BOM_SCALING_TYPE	IN	NUMBER			x
ASSEMBLY_QUANTITY	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
UOM	IN	VARCHAR2(3)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

BILL_SEQUENCE_ID

Bill sequence identifier in the source application instance

ORGANIZATION_ID

Organization identifier of the item

ASSEMBLY_ITEM_ID

Identifier of the item being assembled

ASSEMBLY_TYPE

Manufacturing Bill(1), or Engineering(2). Used for UI and reports.

ALTERNATE_BOM_DESIGNATOR

Name of the bill for alternate bills (null for the primary bill)

SPECIFIC_ASSEMBLY_COMMENT

Comments for specific assembly

PENDING_FROM_ECN

Change notice that created this bill of material

COMMON_BILL_SEQUENCE_ID

Common bill sequence identifier

SCALING_TYPE

Controls scaling behavior

BOM_SCALING_TYPE

BOM scaling type

ASSEMBLY_QUANTITY

Assembly quantity

UOM

Unit of measure code

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_BOM_COMPONENTS

The staging table used by the collection program to valid and process data for table MSC_BOM_COMPONENTS.

Parameter	Usage	Type	Required	Derived	Optional
COMPONENT_SEQUENCE_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
USING_ASSEMBLY_ID	IN	NUMBER			x
BILL_SEQUENCE_ID	IN	NUMBER	x		
COMPONENT_TYPE	IN	NUMBER			x
SCALING_TYPE	IN	NUMBER			x
CHANGE_NOTICE	IN	VARCHAR2(10)			x
REVISION	IN	VARCHAR2(3)			x
UOM_CODE	IN	VARCHAR2(3)			x
USAGE_QUANTITY	IN	NUMBER			x
EFFECTIVITY_DATE	IN	DATE			x
DISABLE_DATE	IN	DATE			x
FROM_UNIT_NUMBER	IN	VARCHAR2(30)			x
TO_UNIT_NUMBER	IN	VARCHAR2(30)			x
USE_UP_CODE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
SUGGESTED_EFFECTIVITY_DATE	IN	DATE			x
DRIVING_ITEM_ID	IN	NUMBER			x
OPERATION_OFFSET_PERCENT	IN	NUMBER			x
OPTIONAL_COMPONENT	IN	NUMBER			x
OLD_EFFECTIVITY_DATE	IN	DATE			x
WIP_SUPPLY_TYPE	IN	NUMBER			x
PLANNING_FACTOR	IN	NUMBER			x
ATP_FLAG	IN	NUMBER			x
COMPONENT_YIELD_FACTOR	IN	NUMBER	x		
REVISED_ITEM_SEQUENCE_ID	IN	NUMBER			x
STATUS_TYPE	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

COMPONENT_SEQUENCE_ID

Component identifier on the source application instance

ORGANIZATION_ID

Organization identifier

INVENTORY_ITEM_ID

Identifier of the component item

USING_ASSEMBLY_ID

Identifier of the item being assembled

BILL_SEQUENCE_ID

Identifier of the BOM

COMPONENT_TYPE

Component (1), Ingredient component (–1), by–product (2)

SCALING_TYPE

Scaling type

CHANGE_NOTICE

Code for ECO. Use for UI and reporting

REVISION

Inventory item revision code

UOM_CODE

Unit of measure code

USAGE_QUANTITY

Quantity of the component to build one unit of item

EFFECTIVITY_DATE

Date of effectivity for this component

DISABLE_DATE

End of effectivity

FROM_UNIT_NUMBER

Effective from this unit number

TO_UNIT_NUMBER

Effective up to this unit number

USE_UP_CODE

Yes/No flag indicating whether the component is effective

SUGGESTED_EFFECTIVITY_DATE

Calculated use–up–date (if Use–up–code is yes)

DRIVING_ITEM_ID

Item which consumption determine the switch to this component

OPERATION_OFFSET_PERCENT

Operation offset percent

OPTIONAL_COMPONENT

Yes/No flag – if optional use planning factor to determine demand

OLD_EFFECTIVITY_DATE

Old effectivity date

WIP_SUPPLY_TYPE

Used mainly for phantoms

PLANNING_FACTOR

Planning factor for this component (percent)

ATP_FLAG

Yes/No flag used for ATP

COMPONENT_YIELD_FACTOR

Factor used to multiply component quantity with to obtain component quantity

REVISED_ITEM_SEQUENCE_ID

Revised item sequence identifier

STATUS_TYPE

Status type

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_BOR_REQUIREMENTS

The staging table used by the collection program to valid and process data for table MSC_BOR_REQUIREMENTS.

Parameter	Usage	Type	Required	Derived	Optional
BILL_OF_RESOURCES	IN	VARCHAR2(10)	x		
ORGANIZATION_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
ASSEMBLY_ITEM_ID	IN	NUMBER	x		
SR_TRANSACTION_ID	IN	NUMBER			x
SOURCE_ITEM_ID	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER			x
RESOURCE_DEPARTMENT_HOURS	IN	NUMBER			x
OPERATION_SEQUENCE_ID	IN	NUMBER			x
OPERATION_SEQ_NUM	IN	NUMBER			x
RESOURCE_SEQ_NUM	IN	NUMBER			x
SETBACK_DAYS	IN	NUMBER			x
DEPARTMENT_ID	IN	NUMBER			x
LINE_ID	IN	NUMBER			x
ASSEMBLY_USAGE	IN	NUMBER			x
ORIGINATION_TYPE	IN	NUMBER			x
RESOURCE_UNITS	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
BASIS	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

BILL_OF_RESOURCES

Bill of resources name

ORGANIZATION_ID

Organization identifier

ASSEMBLY_ITEM_ID

Assembly item identifier

SR_TRANSACTION_ID

Source application transaction identifier

SOURCE_ITEM_ID

Source item identifier

RESOURCE_ID

Resource identifier

RESOURCE_DEPARTMENT_HOURS

Require resource hours

OPERATION_SEQUENCE_ID

Operation sequence identifier

OPERATION_SEQ_NUM

Operation sequence number

RESOURCE_SEQ_NUM

Resource sequence number

SETBACK_DAYS

Resource set back days from assembly due date

DEPARTMENT_ID

Department identifier

LINE_ID

Line identifier

ASSEMBLY_USAGE

Resource hours multiplier for assembly usage

ORIGINATION_TYPE

Load(1), Manual update(2), Manual addition(3)

RESOURCE_UNITS

Operation resource units

BASIS

Operation Basis. Item(1), Lot(2), Resource Units(3), Resource value(4), Total value(5), Activity units(6)

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_CALENDAR_DATES

The staging table used by the collection program to valid and process data for table MSC_CALENDAR_DATES.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_DATE	IN	DATE	x		
CALENDAR_CODE	IN	VARCHAR2(14)	x		
EXCEPTION_SET_ID	IN	NUMBER	x		
SEQ_NUM	IN	NUMBER	x		
NEXT_SEQ_NUM	IN	NUMBER	x		
PRIOR_SEQ_NUM	IN	NUMBER	x		
NEXT_DATE	IN	DATE	x		
PRIOR_DATE	IN	DATE	x		
CALENDAR_START_DATE	IN	DATE	x		
CALENDAR_END_DATE	IN	DATE	x		
DESCRIPTION	IN	VARCHAR2(240)			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

CALENDAR_DATE

Calendar date

CALENDAR_CODE

Calendar code

EXCEPTION_SET_ID

Exception set identifier

SEQ_NUM

Sequence number (for working days only)

NEXT_SEQ_NUM

Next sequence number

PRIOR_SEQ_NUM

Prior sequence number

NEXT_DATE

Date corresponding to next sequence number

PRIOR_DATE

Date corresponding to prior sequence number

CALENDAR_START_DATE

Beginning date for the calendar

CALENDAR_END_DATE

Ending date for the calendar

DESCRIPTION

Calendar description

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_CALENDAR_SHIFTS

The staging table used by the collection program to validate and process data for table MSC_CALENDAR_SHIFTS.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
SHIFT_NUM	IN	NUMBER	x		
DAYS_ON	IN	NUMBER			x
DAYS_OFF	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
DESCRIPTION	IN	VARCHAR2(240)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Calendar code

SHIFT_NUM

Shift number

DAYS_ON

Number of consecutive working days

DAYS_OFF

Number of consecutive non–working days

DESCRIPTION

Description

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_CAL_WEEK_START_DATES

The staging table used by the collection program to validate and process data for table MSC_CAL_WEEK_START_DATES.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
EXCEPTION_SET_ID	IN	NUMBER	x		
WEEK_START_DATE	IN	DATE	x		
NEXT_DATE	IN	DATE	x		
PRIOR_DATE	IN	DATE	x		
SEQ_NUM	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Workday calendar identifier

EXCEPTION_SET_ID

Exception set identifier

WEEK_START_DATE

Week start date

NEXT_DATE

Date corresponding to the next working date

PRIOR_DATE

Date corresponding to the prior working date

SEQ_NUM

Sequence number (for working days)

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_CAL_YEAR_START_DATES

The staging table used by the collection program to validate and process data for table MSC_YEAR_START_DATES.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
EXCEPTION_SET_ID	IN	NUMBER	x		
YEAR_START_DATE	IN	DATE	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Workday calendar identifier

EXCEPTION_SET_ID

Exception set unique identifier

YEAR_START_DATE

Year start date

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_CATEGORY_SETS

The staging table used by the collection program to validate and process data for table MSC_CATEGORY_SETS.

Parameter	Usage	Type	Required	Derived	Optional
CATEGORY_SET_ID	IN	NUMBER			x
SR_CATEGORY_SET_ID	IN	NUMBER	x		
CATEGORY_SET_NAME	IN	VARCHAR2(30)	x		
DESCRIPTION	IN	VARCHAR2(240)			x
CONTROL_LEVEL	IN	NUMBER	x		
DEFAULT_FLAG	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

CATEGORY_SET_ID

Category set identifier

SR_CATEGORY_SET_ID

Category set identifier from source application instance

CATEGORY_SET_NAME

Category set name

DESCRIPTION

Category set description

CONTROL_LEVEL

Control level

DEFAULT_FLAG

Default flag

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh Identifier

MSC_ST_COMPONENT_SUBSTITUTES

The staging table used by the collection program to validate and process data for table MSC_COMPONENT_SUBSTITUTES.

Parameter	Usage	Type	Required	Derived	Optional
COMPONENT_SEQUENCE_ID	IN	NUMBER	x		
SUBSTITUTE_ITEM_ID	IN	NUMBER	x		
USAGE_QUANTITY	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
PRIORITY	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
BILL_SEQUENCE_ID	IN	NUMBER	x		

COMPONENT_SEQUENCE_ID

Component sequence identifier

SUBSTITUTE_ITEM_ID

Substitute item identifier

USAGE_QUANTITY

Usage quantity for the substitute component

ORGANIZATION_ID

Organization identifier

PRIORITY

Priority code

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

BILL_SEQUENCE_ID

Bill sequence identifier

MSC_ST_DEMANDS

The staging table used by the collection program to validate and process data for table MSC_DEMANDS.

Parameter	Usage	Type	Required	Derived	Optional
ORDER_PRIORITY	IN	NUMBER			x
DEMAND_ID	IN	NUMBER			x
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
USING_ASSEMBLY_ITEM_ID	IN	NUMBER	x		
USING_ASSEMBLY_DEMAND_DATE	IN	DATE	x		
USING_REQUIREMENT_QUANTITY	IN	NUMBER	x		
ASSEMBLY_DEMAND_COMP_DATE	IN	DATE			
DEMAND_TYPE	IN	NUMBER	x		
DAILY_DEMAND_RATE	IN	NUMBER			x
ORIGINATION_TYPE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
SOURCE_ORGANIZATION_ID	IN	NUMBER			x
DISPOSITION_ID	IN	NUMBER			x
RESERVATION_ID	IN	NUMBER			x
DEMAND_SCHEDULE_NAME	IN	VARCHAR2(10)			x
PROJECT_ID	IN	NUMBER(15)			x
TASK_ID	IN	NUMBER(15)			x
PLANNING_GROUP	IN	VARCHAR2(30)			x
END_ITEM_UNIT_NUMBER	IN	VARCHAR2(30)			x
SCHEDULE_DATE	IN	DATE			x
OPERATION_SEQ_NUM	IN	NUMBER			x
QUANTITY_ISSUED	IN	NUMBER			x
DEMAND_CLASS	IN	VARCHAR2(34)			x
SALES_ORDER_NUMBER	IN	VARCHAR2(122)			x

Parameter	Usage	Type	Required	Derived	Optional
SALES_ORDER_PRIORITY	IN	NUMBER			x
FORECAST_PRIORITY	IN	NUMBER			x
MPS_DATE_REQUIRED	IN	DATE			x
PO_NUMBER	IN	VARCHAR2(62)			x
WIP_ENTITY_NAME	IN	VARCHAR2(240)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
REPETITIVE_SCHEDULE_ID	IN	NUMBER			x
WIP_ENTITY_ID	IN	NUMBER			x
SELLING_PRICE	IN	NUMBER			x
DMD_LATENESS_COST	IN	NUMBER			x
DMD_SATISFIED_DATE	IN	DATE			x
DMD_SPLIT_FLAG	IN	NUMBER			x
REQUEST_DATE	IN	DATE			x
ORDER_NUMBER	IN	VARCHAR2(240)			x
WIP_STATUS_CODE	IN	NUMBER			x
WIP_SUPPLY_TYPE	IN	NUMBER			x
ATTRIBUTE1	IN	VARCHAR2(150)			x

Parameter	Usage	Type	Required	Derived	Optional
ATTRIBUTE2	IN	VARCHAR2(150)			x
ATTRIBUTE3	IN	VARCHAR2(150)			x
ATTRIBUTE4	IN	VARCHAR2(150)			x
ATTRIBUTE5	IN	VARCHAR2(150)			x
ATTRIBUTE6	IN	VARCHAR2(150)			x
ATTRIBUTE7	IN	VARCHAR2(150)			x
ATTRIBUTE8	IN	VARCHAR2(150)			x
ATTRIBUTE9	IN	VARCHAR2(150)			x
ATTRIBUTE10	IN	VARCHAR2(150)			x
ATTRIBUTE11	IN	VARCHAR2(150)			x
ATTRIBUTE12	IN	VARCHAR2(150)			x
ATTRIBUTE13	IN	VARCHAR2(150)			x
ATTRIBUTE14	IN	VARCHAR2(150)			x
ATTRIBUTE15	IN	VARCHAR2(150)			x

Parameter	Usage	Type	Required	Derived	Optional
SALES_ORDER_LINE_ID	IN	NUMBER			x
CONFIDENCE_PERCENTAGE	IN	NUMBER			x
BUCKET_TYPE	IN	NUMBER			x
BILL_ID	IN	NUMBER			x

ORDER_PRIORITY

Order priority

DEMAND_ID

Demand identifier

INVENTORY_ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

USING_ASSEMBLY_ITEM_ID

Using assembly item identifier (item generates demand for dependent demands)

USING_ASSEMBLY_DEMAND_DATE

Demand date (due date)

USING_REQUIREMENT_QUANTITY

Required quantity

ASSEMBLY_DEMAND_COMP_DATE

Using assembly completion date

DEMAND_TYPE

Discrete Demand(1), Rate–based demand(2)

DAILY_DEMAND_RATE

Repetitive demand rate

ORIGINATION_TYPE

Origin of the demand: Planned order, hard reversation, etc...

SOURCE_ORGANIZATION_ID

Source organization identifier

DISPOSITION_ID

Identifier reference to the supply generating the demand

RESERVATION_ID

Reservation identifier

DEMAND_SCHEDULE_NAME

Demand schedule name

PROJECT_ID

Project identifier to which the demand applies

TASK_ID

Task identifier to which the demand applies

PLANNING_GROUP

Planning group

END_ITEM_UNIT_NUMBER

End item unit number

SCHEDULE_DATE

Schedule date

OPERATION_SEQ_NUM

Operation sequence number

QUANTITY_ISSUED

Quantity issued

DEMAND_CLASS

Demand class code

SALES_ORDER_NUMBER

Sales order number

SALES_ORDER_PRIORITY

Sales order priority

FORECAST_PRIORITY

Forecast priority

MPS_DATE_REQUIRED

MPS date required

PO_NUMBER

Purchase order number

WIP_ENTITY_NAME

Wip job name

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier populated by the collection program

REPETITIVE_SCHEDULE_ID

Repetitive schedule identifier

WIP_ENTITY_ID

WIP job identifier

SELLING_PRICE

Selling price

DMD_LATENESS_COST

Demand lateness cost for independent demands

DMD_SATISFIED_DATE

Date demand is satisfied

DMD_SPLIT_FLAG

Demand split flag

REQUEST_DATE

Request date

ORDER_NUMBER

WIP entity name

WIP_STATUS_CODE

WIP job status code

WIP_SUPPLY_TYPE

WIP supply type

ATTRIBUTE1

Descriptive flexfield segment

ATTRIBUTE2

Descriptive flexfield segment

ATTRIBUTE3

Descriptive flexfield segment

ATTRIBUTE4

Descriptive flexfield segment

ATTRIBUTE5

Descriptive flexfield segment

ATTRIBUTE6

Descriptive flexfield segment

ATTRIBUTE7

Descriptive flexfield segment

ATTRIBUTE8

Descriptive flexfield segment

ATTRIBUTE9

Descriptive flexfield segment

ATTRIBUTE10

Descriptive flexfield segment

ATTRIBUTE11

Descriptive flexfield segment

ATTRIBUTE12

Descriptive flexfield segment

ATTRIBUTE13

Descriptive flexfield segment

ATTRIBUTE14

Descriptive flexfield segment

ATTRIBUTE15

Descriptive flexfield segment

SALES_ORDER_LINE_ID

Sales order line identifier

CONFIDENCE_PERCENTAGE

Forecast confidence percentage

BUCKET_TYPE

Bucket type

BILL_ID

Forecast billing address identifier

MSC_ST_DEMAND_CLASSES

The staging table used by the collection program to validate and process data for demand classes.

Parameter	Usage	Type	Required	Derived	Optional
DEMAND_CLASS	IN	VARCHAR2(30)	x		
MEANING	IN	VARCHAR2(80)	x		
DESCRIPTION	IN	VARCHAR2(250)			x
FROM_DATE	IN	DATE			x
TO_DATE	IN	DATE			x
ENABLED_FLAG	IN	NUMBER	x		
SR_INSTANCE_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)			x
ATTRIBUTE1	IN	VARCHAR2(150)			x
ATTRIBUTE2	IN	VARCHAR2(150)			x
ATTRIBUTE3	IN	VARCHAR2(150)			x
ATTRIBUTE4	IN	VARCHAR2(150)			x

Parameter	Usage	Type	Required	Derived	Optional
ATTRIBUTE5	IN	VARCHAR2(150)			x
ATTRIBUTE6	IN	VARCHAR2(150)			x
ATTRIBUTE7	IN	VARCHAR2(150)			x
ATTRIBUTE8	IN	VARCHAR2(150)			x
ATTRIBUTE9	IN	VARCHAR2(150)			x
ATTRIBUTE10	IN	VARCHAR2(150)			x
ATTRIBUTE11	IN	VARCHAR2(150)			x
ATTRIBUTE12	IN	VARCHAR2(150)			x
ATTRIBUTE13	IN	VARCHAR2(150)			x
ATTRIBUTE14	IN	VARCHAR2(150)			x
ATTRIBUTE15	IN	VARCHAR2(150)			x
DELETED_FLAG	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

DEMAND_CLASS NOT

Demand class code

MEANING NOT

Demand class meaning

DESCRIPTION

Describe the demand class

FROM_DATE

Start date

TO_DATE

End date

ENABLED_FLAG

Enabled flag

SR_INSTANCE_ID NOT

Source application instance identifier

LAST_UPDATE_DATE

Standard Who Column

LAST_UPDATED_BY

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

ATTRIBUTE_CATEGORY

Descriptive flexfield structure defining column

ATTRIBUTE1

Descriptive flexfield segment

ATTRIBUTE2

Descriptive flexfield segment

ATTRIBUTE3

Descriptive flexfield segment

ATTRIBUTE4

Descriptive flexfield segment

ATTRIBUTE5

Descriptive flexfield segment

ATTRIBUTE6

Descriptive flexfield segment

ATTRIBUTE7

Descriptive flexfield segment

ATTRIBUTE8

Descriptive flexfield segment

ATTRIBUTE9

Descriptive flexfield segment

ATTRIBUTE10

Descriptive flexfield segment

ATTRIBUTE11

Descriptive flexfield segment

ATTRIBUTE12

Descriptive flexfield segment

ATTRIBUTE13

Descriptive flexfield segment

ATTRIBUTE14

Descriptive flexfield segment

ATTRIBUTE15

Descriptive flexfield segment

DELETED_FLAG

Deleted flag

REFRESH_ID

Refresh identifier

MSC_ST_DEPARTMENT_RESOURCES

The staging table used by the collection program to validate and process data for table MSC_DEPARTMENT_RESOURCES.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER	x		
RESOURCE_CODE	IN	VARCHAR2(10)			x
DEPARTMENT_ID	IN	NUMBER	x		
DEPARTMENT_CODE	IN	VARCHAR2(10)			x
DEPARTMENT_CLASS	IN	VARCHAR2(10)			x
LINE_FLAG	IN	VARCHAR2(1)	x		

Parameter	Usage	Type	Required	Derived	Optional
OWNING_DEPARTMENT_ID	IN	NUMBER			x
CAPACITY_UNITS	IN	NUMBER			x
MAX_RATE	IN	NUMBER			x
MIN_RATE	IN	NUMBER			x
AGGREGATED_RESOURCE_ID	IN	NUMBER			x
AGGREGATED_RESOURCE_FLAG	IN	NUMBER	x		
RESOURCE_GROUP_NAME	IN	VARCHAR2(30)			x
RESOURCE_GROUP_CODE	IN	VARCHAR2(10)			x
RESOURCE_BALANCE_FLAG	IN	NUMBER			x
BOTTLENECK_FLAG	IN	NUMBER			x
START_TIME	IN	NUMBER			x
STOP_TIME	IN	NUMBER			x
DEPARTMENT_DESCRIPTION	IN	VARCHAR2(240)			x

Parameter	Usage	Type	Required	Derived	Optional
RESOURCE_DESCRIPTION	IN	VARCHAR2(240)			x
OVER_UTILIZED_PERCENT	IN	NUMBER			x
UNDER_UTILIZED_PERCENT	IN	NUMBER			x
RESOURCE_SHORTAGE_TYPE	IN	NUMBER			x
RESOURCE_EXCESS_TYPE	IN	NUMBER			x
USER_TIME_FENCE	IN	NUMBER			x
UTILIZATION	IN	NUMBER			x
EFFICIENCY	IN	NUMBER			x
RESOURCE_INCLUDE_FLAG	IN	NUMBER			x
CRITICAL_RESOURCE_FLAG	IN	NUMBER			x
RESOURCE_TYPE	IN	NUMBER			x
DISABLE_DATE	IN	DATE			x

Parameter	Usage	Type	Required	Derived	Optional
LINE_DISABLE_DATE	IN	DATE			x
AVAILABLE_24_HOURS_FLAG	IN	NUMBER	x		
CTP_FLAG	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
REFRESH_ID	IN	NUMBER			x
DEPT_OVERHEAD_COST	IN	NUMBER			x
RESOURCE_COST	IN	NUMBER			x
RESOURCE_OVER_UTIL_COST	IN	NUMBER			x

ORGANIZATION_ID

Organization identifier

RESOURCE_ID

Source application resource identifier

RESOURCE_CODE

Resource code

DEPARTMENT_ID

Source application department identifier or line identifier

DEPARTMENT_CODE

Department code, also holds line code

DEPARTMENT_CLASS

Department class

LINE_FLAG

Flag to indicate whether or not this resource is a line

OWNING_DEPARTMENT_ID

Owning department identifier

CAPACITY_UNITS

Resource capacity

MAX_RATE

Hourly minimum rate of production line

MIN_RATE

Hourly maximum rate of production line

AGGREGATED_RESOURCE_ID

Reference to aggregate resource, if aggregated

AGGREGATED_RESOURCE_FLAG

Yes/No flag to indicate whether this is an aggregated resource

RESOURCE_GROUP_NAME

Resource group name

RESOURCE_GROUP_CODE

Resource group code

RESOURCE_BALANCE_FLAG

Flag to indicate if the resource needs to load balanced

BOTTLENECK_FLAG

Flag to indicate if the resource is a known bottleneck

START_TIME

Start time of the line

STOP_TIME

Stop time of the line

DEPARTMENT_DESCRIPTION

Describes of the line or department

RESOURCE_DESCRIPTION

Describes the resource

OVER_UTILIZED_PERCENT

Overutilization tolerance

UNDER_UTILIZED_PERCENT

Underutilization tolerance

RESOURCE_SHORTAGE_TYPE

Resource shortage type

RESOURCE_EXCESS_TYPE

Resource excess type

USER_TIME_FENCE

User time fence

UTILIZATION

Utilization

EFFICIENCY

Efficiency

RESOURCE_INCLUDE_FLAG

Resource include flag

CRITICAL_RESOURCE_FLAG

Critical resource flag

RESOURCE_TYPE

Resource type

DISABLE_DATE

Disable date

LINE_DISABLE_DATE

Line disable date

AVAILABLE_24_HOURS_FLAG

Resource is available 24 hours or by shifts

CTP_FLAG

Flag indicating whether the department resource is used for ATP or not

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier populated by the collection program

DEPT_OVERHEAD_COST

Department overhead cost

RESOURCE_COST

Resource cost

RESOURCE_OVER_UTIL_COST

Resource overutilization cost

MSC_ST_DESIGNATORS

The staging table used by the collection program to validate and process data for table MSC_DESIGNATORS.

Parameter	Usage	Type	Required	Derived	Optional
DESIGNATOR_ID	IN	NUMBER	x		
DESIGNATOR	IN	VARCHAR2(10)	x		
SR_DESIGNATOR	IN	VARCHAR2(10)			x
ORGANIZATION_ID	IN	NUMBER	x		
SR_ORGANIZATION_ID	IN	NUMBER			x
MPS_RELIEF	IN	NUMBER	x		
INVENTORY_ATP_FLAG	IN	NUMBER	x		
DESCRIPTION	IN	VARCHAR2(50)			x
DISABLE_DATE	IN	DATE			x
DEMAND_CLASS	IN	VARCHAR2(34)			x
ORGANIZATION_SELECTION	IN	NUMBER			x
PRODUCTION	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
RECOMMENDATION_RELEASE	IN	NUMBER			x
DESIGNATOR_TYPE	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
CONSUME_FORECAST	IN	NUMBER			x
UPDATE_TYPE	IN	NUMBER			x
FORWARD_UPDATE_TIME_FENCE	IN	NUMBER			x
BACKWARD_UPDATE_TIME_FENCE	IN	NUMBER			x
OUTLIER_UPDATE_PERCENTAGE	IN	NUMBER			x
FORECAST_SET_ID	IN	VARCHAR2(10)			x
CUSTOMER_ID	IN	NUMBER			x
SHIP_ID	IN	NUMBER			x
BILL_ID	IN	NUMBER			x
BUCKET_TYPE	IN	NUMBER			x

DESIGNATOR_ID

Designator identifier

DESIGNATOR

Source application schedule name

SR_DESIGNATOR

Source designator identifier

ORGANIZATION_ID

Organization identifier

SR_ORGANIZATION_ID

Source organization identifier

MPS_RELIEF

Flag to indicate whether MPS relief performed against this designator

INVENTORY_ATP_FLAG

ATP supply flag

DESCRIPTION

Description of the this designator

DISABLE_DATE

Designator disable date

DEMAND_CLASS

Demand class code

ORGANIZATION_SELECTION

Single/Multiple organizations

PRODUCTION

Production flag

RECOMMENDATION_RELEASE

Planned order release flag

DESIGNATOR_TYPE

Schedule type

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh number populated by the collection program

CONSUME_FORECAST

Consume forecast flag

UPDATE_TYPE

Forecast update type code

FORWARD_UPDATE_TIME_FENCE

Forward consumption days

BACKWARD_UPDATE_TIME_FENCE

Backward consumption days

OUTLIER_UPDATE_PERCENTAGE

Forecast outlier update percentage

FORECAST_SET_ID

Forecast set identifier

CUSTOMER_ID

Customer identifier

SHIP_ID

Forecast ship code identifier

BILL_ID

Forecast billing address identifier

BUCKET_TYPE

Forecast bucket type – days, weeks or periods

MSC_ST_INTERORG_SHIP_METHODS

The staging table used by the collection program to validate and process data for table MSC_INTERORG_SHIP_METHODS.

Parameter	Usage	Type	Required	Derived	Optional
FROM_ORGANIZATION_ID	IN	NUMBER	x		
TO_ORGANIZATION_ID	IN	NUMBER	x		
SHIP_METHOD	IN	VARCHAR2(30)	x		
TIME_UOM_CODE	IN	VARCHAR2(10)			x
INSTRANSIT_TIME	IN	NUMBER			x
DEFAULT_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
FROM_LOCATION_ID	IN	NUMBER	x		
TO_LOCATION_ID	IN	NUMBER	x		
AVAILABILITY_DATE	IN	DATE			x
WEIGHT_CAPACITY	IN	NUMBER			x
WEIGHT_UOM	IN	VARCHAR2(3)			x
VOLUME_CAPACITY	IN	NUMBER			x
VOLUME_UOM	IN	VARCHAR2(3)			x
COST_PER_WEIGHT_UNIT	IN	NUMBER			x
COST_PER_VOLUME_UNIT	IN	NUMBER			x
INTRANSIT_TIME	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATION_DATE	IN	DATE		x	
CREATED_BY		NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		
TRANSPORT_CAP_OVER_UTIL_COST	IN	NUMBER			x
SR_INSTANCE_ID2	IN	NUMBER	x		

FROM_ORGANIZATION_ID

Organization identifier for the origin organization

TO_ORGANIZATION_ID

Organization identifier for the destination organization'

SHIP_METHOD

Ship method

TIME_UOM_CODE

Unit of measure used to specify the intransit lead time

INSTRANSIT_TIME

Instransit time

DEFAULT_FLAG

Flag to indicate if this is a default ship method

FROM_LOCATION_ID

Location identifier of the origin location

TO_LOCATION_ID

Location identifier of the destination location

AVAILABILITY_DATE

Availability date

WEIGHT_CAPACITY

Weight capacity of this ship method

WEIGHT_UOM

Weight unit of measure

VOLUME_CAPACITY

Weight capacity

VOLUME_UOM

Volume unit of measure

COST_PER_WEIGHT_UNIT

Cost per weight unit

COST_PER_VOLUME_UNIT

Cost per volume unit

INTRANSIT_TIME

Intransit time

DELETED_FLAG

Deleted flag

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier of the source org

TRANSPORT_CAP_OVER_UTIL_COST

Transport cap over utilized cost

SR_INSTANCE_ID2

Source application instance identifier of the destination org

3

Oracle ASCP and Oracle Global ATP Server Open Interfaces - Part 2

This chapter covers the following topics:

- MSC_ST_ITEM_CATEGORIES
- MSC_ST_ITEM_SUPPLIERS
- MSC_ST_LOCATION_ASSOCIATIONS
- MSC_ST_NET_RESOURCE_AVAIL
- MSC_ST_OPERATION_COMPONENTS
- MSC_ST_OPERATION_RESOURCES
- Table and View DefinitionsMSC_ST_OPERATION_RESOURCE_SEQSMSC_ST_OPERATION_RESOURCE_SEQS
- MSC_ST_PARAMETERS
- MSC_ST_PARTNER_CONTACTS
- MSC_ST_PERIOD_START_DATES
- MSC_ST_PLANNERS
- MSC_ST_PROCESS_EFFECTIVITY
- MSC_ST_PROJECTS
- MSC_ST_PROJECT_TASKS
- MSC_ST_RESERVATIONS
- MSC_ST_RESOURCE_CHANGES
- MSC_ST_RESOURCE_GROUPS
- MSC_ST_RESOURCE_REQUIREMENTS
- MSC_ST_RESOURCE_SHIFTS

- MSC_ST_ROUTINGS
- MSC_ST_ROUTING_OPERATIONS
- MSC_ST_SAFETY_STOCKS
- MSC_ST_SALES_ORDERS
- MSC_ST_SHIFT_DATES

MSC_ST_ITEM_CATEGORIES

The staging table used by the collection program to validate and process data for table MSC_ITEM_CATEGORIES.

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
SR_CATEGORY_SET_ID	IN	NUMBER	x		
SR_CATEGORY_ID	IN	NUMBER			x
CATEGORY_NAME	IN	VARCHAR2(163)	x		
DESCRIPTION	IN	VARCHAR2(240)			x
DISABLE_DATE	IN	DATE			x
SUMMARY_FLAG	IN	VARCHAR2(1)	x		
ENABLED_FLAG	IN	VARCHAR2(1)	x		
START_DATE_ACTIVE	IN	DATE			x

Parameter	Usage	Type	Required	Derived	Optional
END_DATE_ACTIVE	IN	DATE			x
CATEGORY_SET_NAME	IN	VARCHAR2(30)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

INVENTORY_ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

SR_CATEGORY_SET_ID

Category set identifier from source application

SR_CATEGORY_ID

Category identifier from source application

CATEGORY_NAME

Category name

DESCRIPTION

Description

DISABLE_DATE

Disable date

SUMMARY_FLAG

Summary flag

ENABLED_FLAG

Enabled flag

START_DATE_ACTIVE

Start date

END_DATE_ACTIVE

End date

CATEGORY_SET_NAME

Category set name

DELETED_FLAG

Deleted flag

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_ITEM_SUPPLIERS

The staging table used by the collection program to validate and process data for table MSC_ITEM_SUPPLIERS.

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
SUPPLIER_ID	IN	NUMBER	x		
SUPPLIER_SITE_ID	IN	NUMBER			x
USING_ORGANIZATION_ID	IN	NUMBER	x		
ASL_ID	IN	NUMBER			x
PROCESSING_LEAD_TIME	IN	NUMBER			x
MINIMUM_ORDER_QUANTITY	IN	NUMBER			x
FIXED_LOT_MULTIPLE	IN	NUMBER			x
DELIVERY_CALENDAR_CODE	IN	VARCHAR2(14)			x
VENDOR_NAME	IN	VARCHAR2(80)			x
VENDOR_SITE_CODE	IN	VARCHAR2(15)			x
SUPPLIER_CAP_OVER_UTIL_COST	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
SR_INSTANCE_ID2	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
PURCHASING_UNIT_OF_MEASURE	IN	VARCHAR2(25)			x

INVENTORY_ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

SUPPLIER_ID

Supplier identifier

SUPPLIER_SITE_ID

Supplier site identifier

USING_ORGANIZATION_ID

Using organization identifier

ASL_ID

ASL identifier

PROCESSING_LEAD_TIME

Processing lead time

MINIMUM_ORDER_QUANTITY

Minimum order quantity

FIXED_LOT_MULTIPLE

Fixed lot multiple

DELIVERY_CALENDAR_CODE

Delivery calendar code

VENDOR_NAME

Supplier name

VENDOR_SITE_CODE

Supplier site code

SUPPLIER_CAP_OVER_UTIL_COST

Supplier cap over util cost

DELETED_FLAG

Deleted flag

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

SR_INSTANCE_ID2

Source application instance identifier of using organization

REFRESH_ID

Refresh identifier

PURCHASING_UNIT_OF_MEASURE

Purchasing unit of measure

MSC_ST_LOCATION_ASSOCIATIONS

The staging table used by the collection program to validate and process data for table MSC_LOCATION_ASSOCIATIONS.

Parameter	Usage	Type	Required	Derived	Optional
LOCATION_ID	IN	NUMBER	x		
SR_INSTANCE_ID	IN	NUMBER	x		
LOCATION_CODE	IN	VARCHAR2(20)			x
ORGANIZATION_ID	IN	NUMBER			x
PARTNER_ID	IN	NUMBER			x
PARTNER_SITE_ID	IN	NUMBER			x
SR_TP_ID	IN	NUMBER	x		
SR_TP_SITE_ID	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
ORGANIZATION_ID	IN	NUMBER			x
REFRESH_ID	IN	NUMBER			x
PARTNER_TYPE	IN	NUMBER	x		

LOCATION_ID

Location identifier

SR_INSTANCE_ID

Source application instance identifier

LOCATION_CODE

Location code

ORGANIZATION_ID

Organization identifier

PARTNER_ID

Partner identifier

PARTNER_SITE_ID

Partner site identifier

SR_TP_ID

Trading partner identifier from source application

SR_TP_SITE_ID

Trading partner site identifier from source application

LAST_UPDATE_DATE

Standard Who Column

LAST_UPDATED_BY

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

ORGANIZATION_ID

Organization identifier

REFRESH_ID

Refresh identifier

PARTNER_TYPE

Partner type

MSC_ST_NET_RESOURCE_AVAIL

The staging table used by the collection program to validate and process data for table MSC_NET_RESOURCE_AVAIL.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
DEPARTMENT_ID	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER	x		
SHIFT_NUM	IN	NUMBER			x
SHIFT_DATE	IN	DATE	x		
FROM_TIME	IN	NUMBER			x
TO_TIME	IN	NUMBER			x
CAPACITY_UNITS	IN	NUMBER	x		
SIMULATION_SET	IN	VARCHAR2(10)			x
AGGREGATE_RESOURCE_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

DEPARTMENT_ID

Department identifier (–1 for lines)

RESOURCE_ID

Resource identifier

SHIFT_NUM

Shift number

SHIFT_DATE

Calendar date

FROM_TIME

Shift start time

TO_TIME

Shift end time

CAPACITY_UNITS

Number of units available during the time interval

SIMULATION_SET

Simulation set identifier

AGGREGATE_RESOURCE_ID

Reference to aggregate resource, if resource aggregated (denormalized column)

DELETED_FLAG

Yes/No flag indicating whether the row will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier populate by the collection program

MSC_ST_OPERATION_COMPONENTS

The staging table used by the collection program to validate and process data for table MSC_OPERATION_COMPONENTS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
OPERATION_SEQUENCE_ID	IN	NUMBER	x		
COMPONENT_SEQUENCE_ID	IN	NUMBER	x		
BILL_SEQUENCE_ID	IN	NUMBER	x		
ROUTING_SEQUENCE_ID	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

ORGANIZATION_ID

Organization identifier

OPERATION_SEQUENCE_ID

Operation sequence identifier

COMPONENT_SEQUENCE_ID

Component sequence identifier

BILL_SEQUENCE_ID

Bill sequence identifier

ROUTING_SEQUENCE_ID

Routing sequence identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS to be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh number populated by the collection program

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_OPERATION_RESOURCES

The staging table used by the collection program to validate and process data for table MSC_OPERATION_RESOURCES.

Parameter	Usage	Type	Required	Derived	Optional
ROUTING_SEQUENCE_ID	IN	NUMBER	x		
RESOURCE_TYPE	IN	NUMBER			x
OPERATION_SEQUENCE_ID	IN	NUMBER	x		
RESOURCE_SEQ_NUM	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER	x		
ALTERNATE_NUMBER	IN	NUMBER	x		
PRINCIPAL_FLAG	IN	NUMBER	x		
BASIS_TYPE	IN	NUMBER	x		
RESOURCE_USAGE	IN	NUMBER	x		
MAX_RESOURCE_UNITS	IN	NUMBER			x
RESOURCE_UNITS	IN	NUMBER			x
UOM_CODE	IN	VARCHAR2(3)	x		

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

ROUTING_SEQUENCE_ID

Routing sequence identifier

RESOURCE_TYPE

Resource type

OPERATION_SEQUENCE_ID

Operation sequence identifier

RESOURCE_SEQ_NUM

Resource sequence number

RESOURCE_ID

Resource identifier

ALTERNATE_NUMBER

Alternate number

PRINCIPAL_FLAG

Flag to indicate whether the resource is the principal resource

BASIS_TYPE

Basis type

RESOURCE_USAGE

Resource usage

MAX_RESOURCE_UNITS

Maximum number of resource units consumed by this operation resource

RESOURCE_UNITS

Operation resource units (capacity)

UOM_CODE

Unit of measure

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh number populated by the collection program

MSC_ST_OPERATION_RESOURCE_SEQS

The staging table used by the collection program to validate and process data for table MSC_OPERATION_RESOURCE_SEQS.

Parameter	Usage	Type	Required	Derived	Optional
ROUTING_SEQUENCE_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
OPERATION_SEQUENCE_ID	IN	NUMBER	x		
RESOURCE_SEQ_NUM	IN	NUMBER	x		
SCHEDULE_FLAG	IN	NUMBER	x		
RESOURCE_OFFSET_PERCENT	IN	NUMBER			x
DEPARTMENT_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

ROUTING_SEQUENCE_ID

Routing sequence identifier

OPERATION_SEQUENCE_ID

Operation sequence identifier

RESOURCE_SEQ_NUM

Resource sequence number

SCHEDULE_FLAG

Schedule

RESOURCE_OFFSET_PERCENT

Resource offset percent

DEPARTMENT_ID

Department identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh number populated by the collection program

MSC_ST_PARAMETERS

The staging table used by the collection program to validate and process data for table MSC_PARAMETERS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
DEMAND_TIME_FENCE_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
PLANNING_TIME_FENCE_FLAG	IN	NUMBER	x		
OPERATION_SCHEDULE_TYPE	IN	NUMBER	x		
CONSIDER_WIP	IN	NUMBER	x		
CONSIDER_PO	IN	NUMBER	x		
SNAPSHOT_LOCK	IN	NUMBER	x		
PLAN_SAFETY_STOCK	IN	NUMBER	x		
CONSIDER_RESERVATIONS	IN	NUMBER	x		
PART_INCLUDE_TYPE	IN	NUMBER	x		
DEFAULT_ABC_ASSIGNMENT_GROUP	IN	VARCHAR2(40)			x
PERIOD_TYPE	IN	NUMBER	x		
RESCHED_ASSUMPTION	IN	NUMBER			x
PLAN_DATE_DEFAULT_TYPE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
INCLUDE_REP_SUPPLY_DAYS	IN	NUMBER			x
INCLUDE_MDS_DAYS	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
REPETITIVE_HORIZON1	IN	NUMBER(38)			x
REPETITIVE_HORIZON2	IN	NUMBER(38)			x
REPETITIVE_BUCKET_SIZE1	IN	NUMBER(38)			x
REPETITIVE_BUCKET_SIZE2	IN	NUMBER(38)			x
REPETITIVE_BUCKET_SIZE3	IN	NUMBER(38)			x
REPETITIVE_ANCHOR_DATE	IN	DATE			x

ORGANIZATION_ID

DEMAND_TIME_FENCE_FLAG

Flag to indicate whether to consider demand time fence

PLANNING_TIME_FENCE_FLAG

IS Flag to indicate whether to consider planning time fence

OPERATION_SCHEDULE_TYPE

Operation schedule type

CONSIDER_WIP

Flag to indicate whether to consider WIP

CONSIDER_PO

Flag to indicate whether to consider PO

SNAPSHOT_LOCK

Flag to indicate whether the snapshot should try to lock tables

PLAN_SAFETY_STOCK

Flag to indicate whether to plan safety stock

CONSIDER_RESERVATIONS

Flag to indicate whether to plan material reservations

PART_INCLUDE_TYPE

Flag to indicate which part to include

DEFAULT_ABC_ASSIGNMENT_GROUP

VARCHAR2(40)

PERIOD_TYPE

Calculate periods based on work dates or calendar dates

RESCHED_ASSUMPTION

Reschedule assumption

PLAN_DATE_DEFAULT_TYPE

Plan date default type

INCLUDE_REP_SUPPLY_DAYS

Flag to indicate whether to include Supply days

INCLUDE_MDS_DAYS

Flag to indicate whether to include MDS days

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier populated by the collection program

REPETITIVE_HORIZON1

First repetitive horizon

REPETITIVE_HORIZON2

Second repetitive horizon

REPETITIVE_BUCKET_SIZE1

First repetitive bucket size

REPETITIVE_BUCKET_SIZE2

Second repetitive bucket size

REPETITIVE_BUCKET_SIZE3

Third repetitive bucket size

REPETITIVE_ANCHOR_DATE

Repetitive anchor date

MSC_ST_PARTNER_CONTACTS

The staging table used by the collection program to validate and process data for table MSC_PARTNER_CONTACTS.

Parameter	Usage	Type	Required	Derived	Optional
NAME	IN	VARCHAR2(100)			x
DISPLAY_NAME	IN	VARCHAR2(240)			x
PARTNER_ID	IN	NUMBER			x
PARTNER_SITE_ID	IN	NUMBER			x
PARTNER_TYPE	IN	NUMBER	x		
EMAIL	IN	VARCHAR2(240)			x
FAX	IN	VARCHAR2(240)			x
ENABLED_FLAG	IN	VARCHAR2(1)			x
DELETED_FLAG	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	

NAME

Partner's user name

DISPLAY_NAME

Partner's display name

PARTNER_ID

Partner Identifier

PARTNER_SITE_ID

Partner site identifier

PARTNER_TYPE

Indicate type of partner, supplier, customer, or buyer

EMAIL

Partner's email address

FAX

Partner's FAX number

ENABLED_FLAG

Flag indicating contact is enabled

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

REFRESH_ID

Refresh ID populated by the pull program

SR_INSTANCE_ID

Source application instance identifier

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

MSC_ST_PERIOD_START_DATES

The staging table used by the collection program to validate and process data for table MSC_PERIOD_START_DATES.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
EXCEPTION_SET_ID	IN	NUMBER	x		
PERIOD_START_DATE	IN	DATE	x		
PERIOD_SEQUENCE_NUM	IN	NUMBER			x
PERIOD_NAME	IN	VARCHAR2(3)			x
NEXT_DATE	IN	DATE	x		
PRIOR_DATE	IN	DATE	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Calendar code

EXCEPTION_SET_ID

Exception set unique identifier

PERIOD_START_DATE

Period start date

PERIOD_SEQUENCE_NUM

Sequence number

PERIOD_NAME

Period Name (depends on quarterly calendar type chosen)

NEXT_DATE

Next calendar date corresponding to next sequence number

PRIOR_DATE

Period start date

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier populated by the collection program

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_PLANNERS

The staging table used by the collection program to validate and process data for table MSC_PLANNERS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
SR_INSTANCE_ID	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE	x		
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
DESCRIPTION	IN	VARCHAR2(50)			x
DISABLE_DATE	IN	DATE			x
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)			x
ATTRIBUTE1	IN	VARCHAR2(150)			x
ATTRIBUTE2	IN	VARCHAR2(150)			x

Parameter	Usage	Type	Required	Derived	Optional
ATTRIBUTE3	IN	VARCHAR2(150)			x
ATTRIBUTE4	IN	VARCHAR2(150)			x
ATTRIBUTE5	IN	VARCHAR2(150)			x
ATTRIBUTE6	IN	VARCHAR2(150)			x
ATTRIBUTE7	IN	VARCHAR2(150)			x
ATTRIBUTE8	IN	VARCHAR2(150)			x
ATTRIBUTE9	IN	VARCHAR2(150)			x
ATTRIBUTE10	IN	VARCHAR2(150)			x
ATTRIBUTE11	IN	VARCHAR2(150)			x
ATTRIBUTE12	IN	VARCHAR2(150)			x
ATTRIBUTE13	IN	VARCHAR2(150)			x
ATTRIBUTE14	IN	VARCHAR2(150)			x
ATTRIBUTE15	IN	VARCHAR2(150)			x
REQUEST_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
ELECTRONIC_MAIL_ADDRESS	IN	VARCHAR2(240)		x	
EMPLOYEE_ID	IN	NUMBER		x	
CURRENT_EMPLOYEE_FLAG	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER			x
USER_NAME	IN	VARCHAR2(100)			x

ORGANIZATION_ID

Organization identifier

SR_INSTANCE_ID

Source application instance identifier

LAST_UPDATE_DATE

Standard Who Column

LAST_UPDATED_BY

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

DESCRIPTION

Describe the planner

DISABLE_DATE

Date on which the planner record is disable

ATTRIBUTE_CATEGORY

Descriptive flexfield structure defining column

ATTRIBUTE1

Descriptive flexfield segment

ATTRIBUTE2

Descriptive flexfield segment

ATTRIBUTE3

Descriptive flexfield segment

ATTRIBUTE4

Descriptive flexfield segment

ATTRIBUTE5

Descriptive flexfield segment

ATTRIBUTE6

Descriptive flexfield segment

ATTRIBUTE7

Descriptive flexfield segment

ATTRIBUTE8

Descriptive flexfield segment

ATTRIBUTE9

Descriptive flexfield segment

ATTRIBUTE10

Descriptive flexfield segment

ATTRIBUTE11

Descriptive flexfield segment

ATTRIBUTE12

Descriptive flexfield segment

ATTRIBUTE13

Descriptive flexfield segment

ATTRIBUTE14

Descriptive flexfield segment

ATTRIBUTE15

Descriptive flexfield segment

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

ELECTRONIC_MAIL_ADDRESS

Electronic mail address

EMPLOYEE_ID

Employee identifier assigned to the planner

CURRENT_EMPLOYEE_FLAG

Flag indicate whether the planner is current employee

REFRESH_ID

Refresh identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

USER_NAME

MSC_ST_PROCESS_EFFECTIVITY

The staging table used by the collection program to validate and process data for table MSC_PROCESS_EFFECTIVITY.

Parameter	Usage	Type	Required	Derived	Optional
PROCESS_SEQUENCE_ID	IN	NUMBER	x		
ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
EFFECTIVITY_DATE	IN	DATE	x		
DISABLE_DATE	IN	DATE			x
MINIMUM_QUANTITY	IN	NUMBER			x
MAXIMUM_QUANTITY	IN	NUMBER			x
PREFERENCE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
ROUTING_SEQUENCE_ID	IN	NUMBER			x
BILL_SEQUENCE_ID	IN	NUMBER			x
TOTAL_PRODUCT_CYCLE_TIME	IN	NUMBER			x
ITEM_PROCESS_COST	IN	NUMBER			x
LINE_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		
PRIMARY_LINE_FLAG	IN	NUMBER			x
PRODUCTION_LINE_RATE	IN	NUMBER			x
LOAD_DISTRIBUTION_PRIORITY	IN	NUMBER			x

PROCESS_SEQUENCE_ID

Process sequence identifier

ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

EFFECTIVITY_DATE

Effectivity date of the process

DISABLE_DATE

Disable date of the process

MINIMUM_QUANTITY

Minimum quantity for which the process can be used to produce the item

MAXIMUM_QUANTITY

Maximum quantity for which the process can be used to produce the item

PREFERENCE

Preference

ROUTING_SEQUENCE_ID

Routing sequence identifier

BILL_SEQUENCE_ID

Bill sequence identifier

TOTAL_PRODUCT_CYCLE_TIME

Total time that an assembly takes along the primary path in the operation network calculated by flow manufacturing

ITEM_PROCESS_COST

Cost of alternate BOM and routing

LINE_ID

Line identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier populated by the collection program

SR_INSTANCE_ID

Source application instance identifier

PRIMARY_LINE_FLAG

Flag indicating whether the line is used for lead time calculations

PRODUCTION_LINE_RATE

Number of assemblies which run down the line per hour

LOAD_DISTRIBUTION_PRIORITY

MSC_ST_PROJECTS

The staging table used by the collection program to validate and process data for table MSC_PROJECTS.

Parameter	Usage	Type	Required	Derived	Optional
PROJECT_ID	IN	NUMBER(15)	x		
ORGANIZATION_ID	IN	NUMBER(15)	x		
PLANNING_GROUP	IN	VARCHAR2(30)			x

Parameter	Usage	Type	Required	Derived	Optional
COSTING_GROUP_ID	IN	NUMBER			x
WIP_ACCT_CLASS_CODE	IN	VARCHAR2(10)			x
SEIBAN_NUMBER_FLAG	IN	NUMBER(1)	x		
PROJECT_NAME	IN	VARCHAR2(30)	x		
PROJECT_NUMBER	IN	VARCHAR2(25)	x		
PROJECT_NUMBER_SORT_ORDER	IN	VARCHAR2(25)			x
PROJECT_DESCRIPTION	IN	VARCHAR2(250)			x
START_DATE	IN	DATE			x
COMPLETION_DATE	IN	DATE			x
OPERATING_UNIT	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
MATERIAL_ACCOUNT	IN	NUMBER			x
MANAGER_CONTACT	IN	VARCHAR2(100)			x

PROJECT_ID

Project identifier or Seiban identifier

ORGANIZATION_ID

Organization identifier

PLANNING_GROUP

Planning group code

COSTING_GROUP_ID

Costing group identifier

WIP_ACCT_CLASS_CODE

Default WIP accounting class assigned to this project

SEIBAN_NUMBER_FLAG

Flag indicates whether project_id identifies a project or a seiban

PROJECT_NAME

Project name

PROJECT_NUMBER

Project number or seiban number

PROJECT_NUMBER_SORT_ORDER

Sort order

PROJECT_DESCRIPTION

Describe the project

START_DATE

Project start date

COMPLETION_DATE

Project completion date

OPERATING_UNIT

Operating unit

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MATERIAL_ACCOUNT

Material account

MANAGER_CONTACT

MSC_ST_PROJECT_TASKS

The staging table used by the collection program to validate and process data for table MSC_PROJECT_TASKS.

Parameter	Usage	Type	Required	Derived	Optional
PROJECT_ID	IN	NUMBER(15)	x		

Parameter	Usage	Type	Required	Derived	Optional
TASK_ID	IN	NUMBER(15)	x		
ORGANIZATION_ID	IN	NUMBER	x		
TASK_NUMBER	IN	VARCHAR2(25)	x		
TASK_NAME	IN	VARCHAR2(20)	x		
DESCRIPTION	IN	VARCHAR2(250)			x
MANAGER	IN	VARCHAR2(240)			x
START_DATE	IN	DATE			x
END_DATE	IN	DATE			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
MANAGER_CONTACT	IN	VARCHAR2(100)			x

PROJECT_ID

Project identifier

TASK_ID

Task identifier

ORGANIZATION_ID

Organization identifier

TASK_NUMBER

Task number

TASK_NAME

Task name

DESCRIPTION

Task description

MANAGER

Manager

START_DATE

Task start date

END_DATE

Task end date

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier populated by the collection program

MSC_ST_RESERVATIONS

The staging table used by the collection program to validate and process data for table MSC_RESERVATIONS.

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
TRANSACTION_ID	IN	NUMBER	x		
PARENT_DEMAND_ID	IN	NUMBER			x
DISPOSITION_ID	IN	NUMBER	x		
REQUIREMENT_DATE	IN	DATE	x		
REVISION	IN	VARCHAR2(3)			x
RESERVED_QUANTITY	IN	NUMBER	x		
DISPOSITION_TYPE	IN	NUMBER	x		
SUBINVENTORY	IN	VARCHAR2(10)			x
RESERVATION_TYPE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
DEMAND_CLASS	IN	VARCHAR2(34)			x
AVAILABLE_TO_MRP	IN	NUMBER			x
RESERVATION_FLAG	IN	NUMBER			x
PROJECT_ID	IN	NUMBER(15)			x
TASK_ID	IN	NUMBER(15)			x
PLANNING_GROUP	IN	VARCHAR2(30)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

INVENTORY_ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

TRANSACTION_ID

Unique identifier generated from the source application instance

PARENT_DEMAND_ID

Parent demand identifier

DISPOSITION_ID

Disposition identifier

REQUIREMENT_DATE

Date of need

REVISION

Inventory item revision code

RESERVED_QUANTITY

Quantity reserved

DISPOSITION_TYPE

Disposition type

SUBINVENTORY

Subinventory identifier

RESERVATION_TYPE

Reservation type

DEMAND_CLASS

Demand class code

AVAILABLE_TO_MRP

Available-to-MRP flag

RESERVATION_FLAG

Reservation flag

PROJECT_ID

Project identifier

TASK_ID

Task identifier

PLANNING_GROUP

Planning group code

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier populated by the collection program

MSC_ST_RESOURCE_CHANGES

The staging table used by the collection program to validate and process data for table MSC_RESOURCE_CHANGES.

Parameter	Usage	Type	Required	Derived	Optional
DEPARTMENT_ID	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER	x		
SHIFT_NUM	IN	NUMBER	x		
FROM_DATE	IN	DATE	x		
TO_DATE	IN	DATE			x
FROM_TIME	IN	NUMBER			x
TO_TIME	IN	NUMBER			x
CAPACITY_CHANGE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
SIMULATION_SET	IN	VARCHAR2(10)	x		
ACTION_TYPE	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

DEPARTMENT_ID

Department identifier (-1 for lines)

RESOURCE_ID

Resource identifier

SHIFT_NUM

Shift number

FROM_DATE

Capacity exception from date

TO_DATE

Capacity exception to date

FROM_TIME

Capacity exception from time

TO_TIME

Capacity exception to time

CAPACITY_CHANGE

Capacity change

SIMULATION_SET

Simulation set identifier

ACTION_TYPE

Type of capacity modification

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier populated by the collection program

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_RESOURCE_GROUPS

The staging table used by the collection program to validate and process data for table MSC_ST_RESOURCE_CHANGES.

Parameter	Usage	Type	Required	Derived	Optional
GROUP_CODE	IN	VARCHAR2(30)	x		
MEANING	IN	VARCHAR2(80)	x		

Parameter	Usage	Type	Required	Derived	Optional
DESCRIPTION	IN	VARCHAR2(250)			x
FROM_DATE	IN	DATE			x
TO_DATE	IN	DATE			x
ENABLED_FLAG	IN	NUMBER	x		
SR_INSTANCE_ID	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)			x
ATTRIBUTE1	IN	VARCHAR2(150)			x
ATTRIBUTE2	IN	VARCHAR2(150)			x
ATTRIBUTE3	IN	VARCHAR2(150)			x
ATTRIBUTE4	IN	VARCHAR2(150)			x
ATTRIBUTE5	IN	VARCHAR2(150)			x
ATTRIBUTE6	IN	VARCHAR2(150)			x
ATTRIBUTE7	IN	VARCHAR2(150)			x
ATTRIBUTE8	IN	VARCHAR2(150)			x
ATTRIBUTE9	IN	VARCHAR2(150)			x
ATTRIBUTE10	IN	VARCHAR2(150)			x
ATTRIBUTE11	IN	VARCHAR2(150)			x
ATTRIBUTE12	IN	VARCHAR2(150)			x
ATTRIBUTE13	IN	VARCHAR2(150)			x

Parameter	Usage	Type	Required	Derived	Optional
ATTRIBUTE14	IN	VARCHAR2(150)			x
ATTRIBUTE15	IN	VARCHAR2(150)			x
DELETED_FLAG	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

GROUP_CODE

Resource group code

MEANING

Meaning

DESCRIPTION

Resource group description

FROM_DATE

Resource start date

TO_DATE

Resource end date

ENABLED_FLAG

Flag indicates whether resource group is enable

SR_INSTANCE_ID

Source application instance identifier

LAST_UPDATE_DATE

Standard Who Column

LAST_UPDATED_BY

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

ATTRIBUTE_CATEGORY

Descriptive flexfield structure defining column

ATTRIBUTE1

Descriptive flexfield segment

ATTRIBUTE2

Descriptive flexfield segment

ATTRIBUTE3

Descriptive flexfield segment

ATTRIBUTE4

Descriptive flexfield segment

ATTRIBUTE5

Descriptive flexfield segment

ATTRIBUTE6

Descriptive flexfield segment

ATTRIBUTE7

Descriptive flexfield segment

ATTRIBUTE8

Descriptive flexfield segment

ATTRIBUTE9

Descriptive flexfield segment

ATTRIBUTE10

Descriptive flexfield segment

ATTRIBUTE11

Descriptive flexfield segment

ATTRIBUTE12

Descriptive flexfield segment

ATTRIBUTE13

Descriptive flexfield segment

ATTRIBUTE14

Descriptive flexfield segment

ATTRIBUTE15

Descriptive flexfield segment

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

REFRESH_ID

Refresh identifier

MSC_ST_RESOURCE_REQUIREMENTS

The staging table used by the collection program to validate and process data for table MSC_RESOURCE_REQUIREMENTS.

Parameter	Usage	Type	Required	Derived	Optional
DEPARTMENT_ID	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
INVENTORY_ITEM_ID	IN	NUMBER			x
SUPPLY_ID	IN	NUMBER			x
OPERATION_SEQ_NUM	IN	NUMBER			x
OPERATION_SEQUENCE_ID	IN	NUMBER			x
RESOURCE_SEQ_NUM	IN	NUMBER	x		
START_DATE	IN	DATE	x		
OPERATION_HOURS_REQUIRED	IN	NUMBER	x		
HOURS_EXPENDED	IN	NUMBER			x
DEMAND_CLASS	IN	VARCHAR2(34)			x
BASIS_TYPE	IN	NUMBER			x
ASSIGNED_UNITS	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
END_DATE	IN	DATE			x
WIP_JOB_TYPE	IN	NUMBER			x
SCHEDULED_COMPLETION_DATE	IN	DATE			x
SCHEDULED_QUANTITY	IN	NUMBER			x
QUANTITY_COMPLETED	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
WIP_ENTITY_ID	IN	NUMBER			x
STD_OP_CODE	IN	VARCHAR2(4)			x
SUPPLY_TYPE	IN	NUMBER			x

DEPARTMENT_ID

Department identifier

RESOURCE_ID

Resource identifier

ORGANIZATION_ID

Organization identifier

INVENTORY_ITEM_ID

Inventory item identifier

SUPPLY_ID

Supply identifier

OPERATION_SEQ_NUM

Operation sequence number

OPERATION_SEQUENCE_ID

Operation sequence identifier

RESOURCE_SEQ_NUM

Resource sequence number

START_DATE

Start date of the resource requirement

OPERATION_HOURS_REQUIRED

Operation hours required

HOURS_EXPENDED

Hours expended

DEMAND_CLASS

Demand class code

BASIS_TYPE

Basis type

ASSIGNED_UNITS

Assigned units

END_DATE

End date of the resource requirement

WIP_JOB_TYPE

WIP job type

SCHEDULED_COMPLETION_DATE

Schedule completion date

SCHEDULED_QUANTITY

Quantity scheduled

QUANTITY_COMPLETED

Quantity completed

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

WIP_ENTITY_ID

WIP job identifier

STD_OP_CODE

Standard OP code

SUPPLY_TYPE

Supply type

MSC_ST_RESOURCE_SHIFTS

The staging table used by the collection program to validate and process data for table MSC_RESOURCE_SHIFTS.

Parameter	Usage	Type	Required	Derived	Optional
DEPARTMENT_ID	IN	NUMBER	x		
RESOURCE_ID	IN	NUMBER	x		
SHIFT_NUM	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

DEPARTMENT_ID

Department identifier

RESOURCE_ID

Resource identifier

SHIFT_NUM

Shift number

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_ROUTINGS

The staging table used by the collection program to validate and process data for table MSC_ROUTINGS.

Parameter	Usage	Type	Required	Derived	Optional
ROUTING_SEQUENCE_ID	IN	NUMBER	x		
ASSEMBLY_ITEM_ID	IN	NUMBER	x		
ROUTING_TYPE	IN	NUMBER	x		
ROUTING_COMMENT	IN	VARCHAR2(240)			x
PRIORITY	IN	NUMBER			x
ALTERNATE_ROUTING_DESIGNATOR	IN	VARCHAR2(10)			x

Parameter	Usage	Type	Required	Derived	Optional
PROJECT_ID	IN	NUMBER			x
TASK_ID	IN	NUMBER			x
LINE_ID	IN	NUMBER			x
UOM_CODE	IN	VARCHAR2(3)			x
CFM_ROUTING_FLAG	IN	NUMBER			x
CTP_FLAG	IN	NUMBER			x
ROUTING_QUANTITY	IN	NUMBER			x
COMPLETION_SUBINVENTORY	IN	VARCHAR2(10)			x
COMPLETION_LOCATOR_ID	IN	NUMBER			x
COMMON_ROUTING_SEQUENCE_ID	IN	NUMBER			x
MIXED_MODEL_MAP_FLAG	IN	NUMBER			x
TOTAL_PRODUCT_CYCLE_TIME	IN	NUMBER			x
ORGANIZATION_ID	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

ROUTING_SEQUENCE_ID

Routing sequence identifier

ASSEMBLY_ITEM_ID

Assembly item identifier

ROUTING_TYPE

Routing type

ROUTING_COMMENT

Routing comment

PRIORITY

Routing priority

ALTERNATE_ROUTING_DESIGNATOR

Name of the alternate routing. Null for primary routing

PROJECT_ID

Project identifier

TASK_ID

Task identifier

LINE_ID

Manufacturing line identifier

UOM_CODE

Unit of measure code

CFM_ROUTING_FLAG

CFM routing flag

CTP_FLAG

CTP flag

ROUTING_QUANTITY

Routing quantity

COMPLETION_SUBINVENTORY

Completion subinventory

COMPLETION_LOCATOR_ID

Completion locator identifier

COMMON_ROUTING_SEQUENCE_ID

Common routing sequence identifier

MIXED_MODEL_MAP_FLAG

Mix model map flag

TOTAL_PRODUCT_CYCLE_TIME

Total product cycle time

ORGANIZATION_ID

Organization identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_ROUTING_OPERATIONS

The staging table used by the collection program to validate and process data for table MSC_ROUTING_OPERATIONS.

Parameter	Usage	Type	Required	Derived	Optional
OPERATION_SEQUENCE_ID	IN	NUMBER	x		
ROUTING_SEQUENCE_ID	IN	NUMBER	x		
OPERATION_SEQ_NUM	IN	NUMBER	x		
OPERATION_DESCRIPTION	IN	VARCHAR2(240)			x
EFFECTIVITY_DATE	IN	DATE	x		
DISABLE_DATE	IN	DATE			x
FROM_UNIT_NUMBER	IN	VARCHAR2(30)			x
TO_UNIT_NUMBER	IN	VARCHAR2(30)			x
OPTION_DEPENDENT_FLAG	IN	NUMBER	x		
OPERATION_TYPE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
MINIMUM_TRANSFER_QUANTITY	IN	NUMBER			x
YIELD	IN	NUMBER			x
DEPARTMENT_ID	IN	NUMBER	x		
DEPARTMENT_CODE	IN	VARCHAR2(10)			x
OPERATION_LEAD_TIME_PERCENT	IN	NUMBER			x
CUMULATIVE_YIELD	IN	NUMBER			x
REVERSE_CUMULATIVE_YIELD	IN	NUMBER			x
NET_PLANNING_PERCENT	IN	NUMBER			x
TEAR_DOWN_DURATION	IN	NUMBER			x
SETUP_DURATION	IN	NUMBER			x
UOM_CODE	IN	VARCHAR2(3)			x
STANDARD_OPERATION_CODE	IN	VARCHAR2(4)			x

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

OPERATION_SEQUENCE_ID

Operation sequence identifier

ROUTING_SEQUENCE_ID

Routing sequence identifier

OPERATION_SEQ_NUM

Operation sequence number

OPERATION_DESCRIPTION

Operation description

EFFECTIVITY_DATE

Date operation is effective

DISABLE_DATE

End of effectivity

FROM_UNIT_NUMBER

Effective from this unit number

TO_UNIT_NUMBER

Effective up to this unit number

OPTION_DEPENDENT_FLAG

Flag to indicate whether this operation option dependent

OPERATION_TYPE

Indicate operation type: Process, Line, or Event.

MINIMUM_TRANSFER_QUANTITY

Minimum operation transfer quantity

YIELD

Process yield at this operation

DEPARTMENT_ID

Department identifier

DEPARTMENT_CODE

Department code

OPERATION_LEAD_TIME_PERCENT

Indicates the amount of overlap its lead time has with the parent lead time

CUMULATIVE_YIELD

Cumulative process yield from the beginning of routing to this operation

REVERSE_CUMULATIVE_YIELD

Cumulative process yield from the end of routing to comparable operation

NET_PLANNING_PERCENT

Cumulative planning percents derived from the operation network

TEAR_DOWN_DURATION

Duration of the tear down for this operation

SETUP_DURATION

Duration of the set-up

UOM_CODE

Unit of measure code

STANDARD_OPERATION_CODE

Code of the standard operation on which this operation is based

ORGANIZATION_ID

Organization identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_SAFETY_STOCKS

The staging table used by the collection program to validate and process data for table MSC_SAFETY_STOCKS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
INVENTORY_ITEM_ID	IN	NUMBER	x		
PERIOD_START_DATE	IN	DATE	x		
SAFETY_STOCK_QUANTITY	IN	NUMBER	x		
UPDATED	IN	NUMBER			x
STATUS	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

ORGANIZATION_ID

Organization identifier

INVENTORY_ITEM_ID

Inventory item identifier

PERIOD_START_DATE

Period start date

SAFETY_STOCK_QUANTITY

Safety stock quantity

UPDATED

Updated flag

STATUS

Status flag

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_SALES_ORDERS

The staging table used by the collection program to validate and process data for table MSC_SAFETY_STOCKS.

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
DEMAND_ID	IN	NUMBER	x		
PRIMARY_UOM_QUANTITY	IN	NUMBER	x		
RESERVATION_TYPE	IN	NUMBER			x
RESERVATION_QUANTITY	IN	NUMBER			x
DEMAND_SOURCE_TYPE	IN	NUMBER	x		
DEMAND_SOURCE_HEADER_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
COMPLETED_QUANTITY	IN	NUMBER	x		
SUBINVENTORY	IN	VARCHAR2(10)			x
DEMAND_CLASS	IN	VARCHAR2(34)			x
REQUIREMENT_DATE	IN	DATE	x		
DEMAND_SOURCE_LINE	IN	VARCHAR2(40)			x
DEMAND_SOURCE_DELIVERY	IN	VARCHAR2(30)			x
DEMAND_SOURCE_NAME	IN	VARCHAR2(30)			x
PARENT_DEMAND_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		
SALES_ORDER_NUMBER	IN	VARCHAR2(122)			x
SALESREP_CONTACT	IN	VARCHAR2(100)			x
ORDERED_ITEM_ID	IN	NUMBER			x
AVAILABLE_TO_MRP	IN	VARCHAR2(1)			x
CUSTOMER_ID	IN	NUMBER			x
SHIP_TO_SITE_USE_ID	IN	NUMBER			x
BILL_TO_SITE_USE_ID	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
LINE_NUM	IN	NUMBER			x
TERRITORY_ID	IN	NUMBER			x
UPDATE_SEQ_NUM	IN	NUMBER			x
DEMAND_TYPE	IN	NUMBER			x
PROJECT_ID	IN	NUMBER			x
TASK_ID	IN	NUMBER			x
PLANNING_GROUP	IN	VARCHAR2(30)			x
END_ITEM_UNIT_NUMBER	IN	VARCHAR2(30)			x
DEMAND_PRIORITY	IN	NUMBER			x

INVENTORY_ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

DEMAND_ID

Unique identifier of a demand row from source application instance

PRIMARY_UOM_QUANTITY

Primary UOM quantity

RESERVATION_TYPE

Code for type of reservation

RESERVATION_QUANTITY

Total quantity reserved expressed in primary unit of measure

DEMAND_SOURCE_TYPE

Demand source type

DEMAND_SOURCE_HEADER_ID

Header ID for the source of the demand

COMPLETED_QUANTITY

Completed quantity

SUBINVENTORY

Subinventory code

DEMAND_CLASS

Demand class code

REQUIREMENT_DATE

Planned ship date for summary demand

DEMAND_SOURCE_LINE

Line id of demand source

DEMAND_SOURCE_DELIVERY

For Sales Order demand, Line id of Sales order line detail row (SO_LINE_DETAILS.LINE_DETAIL_I D) from source application instance

DEMAND_SOURCE_NAME

Identifier for user-defined Source Type

PARENT_DEMAND_ID

Parent demand identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh number populated by the collection program

SR_INSTANCE_ID

Source application instance identifier

SALES_ORDER_NUMBER

Sales order number

SALESREP_CONTACT

ORDERED_ITEM_ID

Ordered item identifier

AVAILABLE_TO_MRP

Available to MRP flag

CUSTOMER_ID

Customer identifier

SHIP_TO_SITE_USE_ID

Ship to identifier of the sales order

BILL_TO_SITE_USE_ID

Bill to identifier of the sales order

LINE_NUM

Sales order line number

TERRITORY_ID

Territory identifier of the sales order

UPDATE_SEQ_NUM

Update sequence number

DEMAND_TYPE

Demand type

PROJECT_ID

Project identifier

TASK_ID

Task identifier

PLANNING_GROUP

Planning group

END_ITEM_UNIT_NUMBER

Unit number identifier

DEMAND_PRIORITY

Demand priority

MSC_ST_SHIFT_DATES

The staging table used by the collection program to validate and process data for table MSC_SHIFT_DATES.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
EXCEPTION_SET_ID	IN	NUMBER	x		
SHIFT_NUM	IN	NUMBER	x		
SHIFT_DATE	IN	DATE	x		
SEQ_NUM	IN	NUMBER			x
NEXT_SEQ_NUM	IN	NUMBER	x		
PRIOR_SEQ_NUM	IN	NUMBER	x		
NEXT_DATE	IN	DATE	x		
PRIOR_DATE	IN	DATE	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Calendar code

EXCEPTION_SET_ID

Exception set identifier

SHIFT_NUM

Calendar shift number

SHIFT_DATE

Calendar date

SEQ_NUM

Sequence number for shift date (only for working dates)

NEXT_SEQ_NUM

Next sequence number for calendar date (working day)

PRIOR_SEQ_NUM

Prior sequence number for calendar date (working day)

NEXT_DATE

Next date corresponding to next sequence number

PRIOR_DATE

Prior date corresponding to prior sequence number

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

4

Oracle ASCP and Oracle Global ATP Server Open Interfaces - Part 3

This chapter covers the following topics:

- MSC_ST_SHIFT_EXCEPTIONS
- MSC_ST_SHIFT_TIMES
- MSC_ST_SIMULATION_SETS
- MSC_ST_SOURCING_HISTORY
- MSC_ST_SOURCING_RULES
- MSC_ST_SR_ASSIGNMENTS
- MSC_ST_SR_RECEIPT_ORG
- MSC_ST_SR_SOURCE_ORG
- MSC_ST_SUB_INVENTORIES
- MSC_ST_SUPPLIER_CAPACITIES
- MSC_ST_SUPPLIES
- MSC_ST_SYSTEM_ITEMS
- MSC_ST_TRADING_PARTNERS
- MSC_ST_TRADING_PARTNER_SITES
- MSC_ST_UNITS_OF_MEASURE
- MSC_ST_UOM_CLASS_CONVERSIONS
- MSC_ST_UOM_CONVERSIONS

MSC_ST_SHIFT_EXCEPTIONS

The staging table used by the collection program to validate and process data for table

MSC_SHIFT_EXCEPTIONS.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
SHIFT_NUM	IN	NUMBER	x		
EXCEPTION_SET_ID	IN	NUMBER	x		
EXCEPTION_DATE	IN	DATE	x		
EXCEPTION_TYPE	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Calendar code

SHIFT_NUM

Calendar shift number

EXCEPTION_SET_ID

Exception set identifier

EXCEPTION_DATE

Exception date

EXCEPTION_TYPE

Exception type

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_SHIFT_TIMES

The staging table used by the collection program to validate and process data for table MSC_SHIFT_TIMES.

Parameter	Usage	Type	Required	Derived	Optional
CALENDAR_CODE	IN	VARCHAR2(14)	x		
SHIFT_NUM	IN	NUMBER	x		
FROM_TIME	IN	NUMBER	x		
TO_TIME	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
REFRESH_ID	IN	NUMBER			x
SR_INSTANCE_ID	IN	NUMBER	x		

CALENDAR_CODE

Calendar code

SHIFT_NUM

Shift number

FROM_TIME

Shift start time

TO_TIME

Shift end time

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

REFRESH_ID

Refresh identifier

SR_INSTANCE_ID

Source application instance identifier

MSC_ST_SIMULATION_SETS

The staging table used by the collection program to validate and process data for table MSC_SIMULATION_SETS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
SIMULATION_SET	IN	VARCHAR2(10)	x		
DESCRIPTION	IN	VARCHAR2(50)			x
USE_IN_WIP_FLAG	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

ORGANIZATION_ID

Organization identifier

SIMULATION_SET

Simulation set

DESCRIPTION

Describe simulation set

USE_IN_WIP_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

DELETED_FLAG

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_SOURCING_HISTORY

The staging table used by the collection program to validate and process data for table MSC_SOURCING_HISTORY.

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
SR_INSTANCE_ID	IN	NUMBER	x		
SOURCING_RULE_ID	IN	NUMBER	x		
SOURCE_ORG_ID	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
SOURCE_SR_INSTANCE_ID	IN	NUMBER			x
SUPPLIER_ID	IN	NUMBER			x
SUPPLIER_SITE_ID	IN	NUMBER			x
HISTORICAL_ALLOCATION	IN	NUMBER	x		
REFRESH_NUMBER	IN	NUMBER			x
LAST_CALCULATED_DATE	IN	DATE			x
LAST_UPDATED_BY	IN	NUMBER		x	
LAST_UPDATE_DATE	IN	DATE		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	

Index Name	Index Type	Sequence	Column Name
MSC_ST_SOURCING_HISTORY_U1	UNIQUE	1	SOURCING_RULE_ID
		2	INVENTORY_ITEM_ID
		3	ORGANIZATION_ID
		4	SR_INSTANCE_ID

INVENTORY_ITEM_ID

Inventory Item Id

ORGANIZATION_ID

Organization Id

SR_INSTANCE_ID

sr instance Id

SOURCING_RULE_ID

Sourcing Rule/Bill of Distribution identifier

SOURCE_ORG_ID

Source Org Id

SOURCE_SR_INSTANCE_ID

source org sr instance Id

SUPPLIER_ID

Supplier identifier

SUPPLIER_SITE_ID

Supplier site identifier

HISTORICAL_ALLOCATION

Historical Allocation

REFRESH_NUMBER

Refresh Number

LAST_CALCULATED_DATE

Last Calculated Date

LAST_UPDATED_BY

Standard Who Column

LAST_UPDATE_DATE

Standard Who Column

CREATION_DATE

Standard Who Column

CREATED_BY

Standard Who Column

LAST_UPDATE_LOGIN

Standard Who Column

REQUEST_ID

Concurrent Who Column

PROGRAM_APPLICATION_ID

Concurrent Who Column

PROGRAM_ID

Concurrent Who Column

PROGRAM_UPDATE_DATE

Concurrent Who Column

MSC_ST_SOURCING_RULES

The staging table used by the collection program to validate and process data for table MSC_SOURCING_RULES.

Parameter	Usage	Type	Required	Derived	Optional
SOURCING_RULE_ID	IN	NUMBER			x
SR_SOURCING_RULE_ID	IN	NUMBER	x		
SOURCING_RULE_NAME	IN	VARCHAR2(30)	x		
ORGANIZATION_ID	IN	NUMBER			x
DESCRIPTION	IN	VARCHAR2(80)			x
STATUS	IN	NUMBER	x		
SOURCING_RULE_TYPE	IN	NUMBER	x		
PLANNING_ACTIVE	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

SOURCING_RULE_ID

Sourcing rule / Bill of Distribution identifier

SR_SOURCING_RULE_ID

Sourcing rule / Bill of Distribution identifier from source application

SOURCING_RULE_NAME

Sourcing rule / Bill of Distribution name

ORGANIZATION_ID

Organization identifier

DESCRIPTION

Describe Sourcing rule / Bill of Distribution

STATUS

Status flag

SOURCING_RULE_TYPE

Flag indicates whether the row is sourcing rule or bill of distribution

PLANNING_ACTIVE

Flag indicates whether the row is planning active

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_SR_ASSIGNMENTS

The staging table used by the collection program to validate and process data for table MSC_SR_ASSIGNMENTS.

Parameter	Usage	Type	Required	Derived	Optional
ASSIGNMENT_ID	IN	NUMBER			x
SR_ASSIGNMENT_ID	IN	NUMBER	x		
ASSIGNMENT_SET_ID	IN	NUMBER	x		
ASSIGNMENT_TYPE	IN	NUMBER	x		
SOURCING_RULE_ID	IN	NUMBER	x		
SOURCING_RULE_TYPE	IN	NUMBER			x
INVENTORY_ITEM_ID	IN	NUMBER			x
PARTNER_ID	IN	NUMBER			x
SHIP_TO_SITE_ID	IN	NUMBER			x
CUSTOMER_NAME	IN	VARCHAR2(50)			x
SITE_USE_CODE	IN	VARCHAR2(30)			x

Parameter	Usage	Type	Required	Derived	Optional
LOCATION	IN	VARCHAR2(40)			x
ORGANIZATION_ID	IN	NUMBER			x
CATEGORY_ID	IN	NUMBER			x
CATEGORY_NAME	IN	VARCHAR2(163)			x
CATEGORY_SET_IDENTIFIER	IN	NUMBER			x
CATEGORY_SET_NAME	IN	VARCHAR2(30)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
SR_ASSIGNMENT_INSTANCE_ID	IN	NUMBER			x

ASSIGNMENT_ID

Unique identifier for the row

SR_ASSIGNMENT_ID

Unique identifier for the row from the source application

ASSIGNMENT_SET_ID

Assignment set unique identifier

ASSIGNMENT_TYPE

Assignment set type

SOURCING_RULE_ID

Sourcing rule / Bill of Distribution identifier

SOURCING_RULE_TYPE

Sourcing rule type

INVENTORY_ITEM_ID

Inventory item identifier

PARTNER_ID

Trading partner identifier

SHIP_TO_SITE_ID

Ship to site identifier

CUSTOMER_NAME

Customer name

SITE_USE_CODE

Site use code

LOCATION

Location

ORGANIZATION_ID

Organization identifier

CATEGORY_ID

Category identifier

CATEGORY_NAME

Category name

CATEGORY_SET_IDENTIFIER

Category set identifier

CATEGORY_SET_NAME

Category set name

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

SR_ASSIGNMENT_INSTANCE_ID

Source application instance identifier for the source assignment record

MSC_ST_SR_RECEIPT_ORG

The staging table used by the collection program to validate and process data for table MSC_SR_RECEIPT_ORG.

Parameter	Usage	Type	Required	Derived	Optional
SR_RECEIPT_ID	IN	NUMBER	x		
SR_SR_RECEIPT_ORG	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
SOURCING_RULE_ID	IN	NUMBER	x		
RECEIPT_PARTNER_ID	IN	NUMBER			x
RECEIPT_PARTNER_SITE_ID	IN	NUMBER			x
EFFECTIVE_DATE	IN	DATE	x		
DISABLE_DATE	IN	DATE			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
RECEIPT_ORG_INSTANCE_ID	IN	NUMBER			x

SR_RECEIPT_ID

Unique identifier for a row generated at planning server

SR_SR_RECEIPT_ORG

Receiving org from source application instance

SOURCING_RULE_ID

Sourcing rule / Bill of Distribution identifier

RECEIPT_PARTNER_ID

Trading partner unique identifier

RECEIPT_PARTNER_SITE_ID

Trading partner site unique identifier

EFFECTIVE_DATE

Date of effectivity

DISABLE_DATE

Disable date

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

RECEIPT_ORG_INSTANCE_ID

Source application instance identifier associated with the receiving org

MSC_ST_SR_SOURCE_ORG

The staging table used by the collection program to validate and process data for table MSC_SR_SOURCE_ORG.

Parameter	Usage	Type	Required	Derived	Optional
SR_SOURCE_ID	IN	NUMBER			x
SR_SR_SOURCE_ID	IN	NUMBER	x		
SR_RECEIPT_ID	IN	NUMBER	x		
SOURCE_ORGANIZATION_ID	IN	NUMBER			x
SOURCE_PARTNER_ID	IN	NUMBER			x
SOURCE_PARTNER_SITE_ID	IN	NUMBER			x
SECONDARY_INVENTORY	IN	VARCHAR2(10)			x
SOURCE_TYPE	IN	NUMBER			x
ALLOCATION_PERCENT	IN	NUMBER	x		
RANK	IN	NUMBER			x
VENDOR_NAME	IN	VARCHAR2(80)			x
VENDOR_SITE_CODE	IN	VARCHAR2(15)			x
SHIP_METHOD	IN	VARCHAR2(30)			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
SOURCE_ORG_INSTANCE_ID	IN	NUMBER			x

SR_SOURCE_ID

Unique identifier for a row generated at planning server

SR_SR_SOURCE_ID

Unique identifier for the row generated at the source application

SR_RECEIPT_ID

SR receipt unique identifier

SOURCE_ORGANIZATION_ID

Source organization identifier

SOURCE_PARTNER_ID

Source trading partner identifier

SOURCE_PARTNER_SITE_ID

Source trading partner site identifier

SECONDARY_INVENTORY

Secondary inventory code (not currently used)

SOURCE_TYPE

Source type

ALLOCATION_PERCENT

Percent of supply allocated to this source

RANK

Rank of source

VENDOR_NAME

Supplier name

VENDOR_SITE_CODE

Supplier site code

SHIP_METHOD

Ship method

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier associated with the sr source org record

REFRESH_ID

Refresh identifier

SOURCE_ORG_INSTANCE_ID

Source application instance identifier associated with the source organization

MSC_ST_SUB_INVENTORIES

The staging table used by the collection program to validate and process data for table MSC_SUB_INVENTORIES.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
SUB_INVENTORY_CODE	IN	VARCHAR2(10)	x		
DESCRIPTION	IN	VARCHAR2(50)			x
DISABLE_DATE	IN	DATE			x
NETTING_TYPE	IN	NUMBER			x
DEMAND_CLASS	IN	VARCHAR2(34)			x
PROJECT_ID	IN	NUMBER(15)			x
TASK_ID	IN	NUMBER(15)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
INVENTORY_ATP_CODE	IN	NUMBER			x

ORGANIZATION_ID

Organization identifier

SUB_INVENTORY_CODE

Sub-inventory code

DESCRIPTION

Describe sub-inventory

DISABLE_DATE

Date on which the row is no longer in used

NETTING_TYPE

Netting type

DEMAND_CLASS

Demand class code

PROJECT_ID

Project identifier

TASK_ID

Task identifier

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

INVENTORY_ATP_CODE

Inventory ATP code

MSC_ST_SUPPLIER_CAPACITIES

The staging table used by the collection program to validate and process data for table MSC_SUPPLIER_CAPACITIES.

Parameter	Usage	Type	Required	Derived	Optional
SUPPLIER_ID	IN	NUMBER	x		
SUPPLIER_SITE_ID	IN	NUMBER			x
ORGANIZATION_ID	IN	NUMBER	x		
USING_ORGANIZATION_ID	IN	NUMBER	x		
INVENTORY_ITEM_ID	IN	NUMBER	x		
VENDOR_NAME	IN	VARCHAR2(80)			x
VENDOR_SITE_CODE	IN	VARCHAR2(15)			x
FROM_DATE	IN	DATE	x		
TO_DATE	IN	DATE			x
CAPACITY	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

SUPPLIER_ID

Supplier identifier

SUPPLIER_SITE_ID

Supplier site identifier

ORGANIZATION_ID

Organization identifier

USING_ORGANIZATION_ID

Using organization identifier

INVENTORY_ITEM_ID

Inventory item identifier

VENDOR_NAME

Supplier name

VENDOR_SITE_CODE

Supplier site code

FROM_DATE

First date of valid capacity

TO_DATE

Last date of valid capacity

CAPACITY

Capacity

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_SUPPLIER_FLEX_FENCES

The staging table used by the collection program to validate and process data for table MSC_SUPPLIER_FLEX_FENCES.

Parameter	Usage	Type	Required	Derived	Optional
SUPPLIER_ID	IN	NUMBER	x		
SUPPLIER_SITE_ID	IN	NUMBER			x
ORGANIZATION_ID	IN	NUMBER	x		
USING_ORGANIZATION_ID	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
VENDOR_NAME	IN	VARCHAR2(80)			x
VENDOR_SITE_CODE	IN	VARCHAR2(15)			x
FENCE_DAYS	IN	NUMBER	x		
TOLERANCE_PERCENTAGE	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

SUPPLIER_ID

Supplier identifier

SUPPLIER_SITE_ID

Supplier site identifier

ORGANIZATION_ID

Organization identifier

USING_ORGANIZATION_ID

Using organization identifier

INVENTORY_ITEM_ID

Inventory item identifier

VENDOR_NAME

Supplier name

VENDOR_SITE_CODE

Supplier site code

FENCE_DAYS

Number of advance days

TOLERANCE_PERCENTAGE

Capacity tolerance percentage

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_SUPPLIES

The staging table used by the collection program to validate and process data for table MSC_SUPPLIES.

Parameter	Usage	Type	Required	Derived	Optional
PLAN_ID	IN	NUMBER			x
TRANSACTION_ID	IN	NUMBER			x
INVENTORY_ITEM_ID	IN	NUMBER	x		
ORGANIZATION_ID	IN	NUMBER	x		
SCHEDULE_DESIGNATOR_ID	IN	NUMBER			x
SOURCE_SCHEDULE_NAME	IN	VARCHAR2(10)			x
REVISION	IN	VARCHAR2(10)			x
UNIT_NUMBER	IN	VARCHAR2(30)			x
NEW_SCHEDULE_DATE	IN	DATE	x		
OLD_SCHEDULE_DATE	IN	DATE			x
NEW_WIP_START_DATE	IN	DATE			x
OLD_WIP_START_DATE	IN	DATE			x
FIRST_UNIT_COMPLETION_DATE	IN	DATE			x

Parameter	Usage	Type	Required	Derived	Optional
LAST_UNIT_COMPLETION_DATE	IN	DATE			x
FIRST_UNIT_START_DATE	IN	DATE			x
LAST_UNIT_START_DATE	IN	DATE			x
DISPOSITION_ID	IN	NUMBER			x
DISPOSITION_STATUS_TYPE	IN	NUMBER			x
ORDER_TYPE	IN	NUMBER	x		
SUPPLIER_ID	IN	NUMBER			x
NEW_ORDER_QUANTITY	IN	NUMBER	x		
OLD_ORDER_QUANTITY	IN	NUMBER			x
NEW_ORDER_PLACEMENT_DATE	IN	DATE			x
OLD_ORDER_PLACEMENT_DATE	IN	DATE			x
RESCHEDULE_DAYS	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
RESCHEDULE_FLAG	IN	NUMBER			x
SCHEDULE_COMPRESS_DAYS	IN	NUMBER			x
NEW_PROCESSING_DAYS	IN	NUMBER			x
PURCH_LINE_NUM	IN	NUMBER			x
QUANTITY_IN_PROCESS	IN	NUMBER			x
IMPLEMENTED_QUANTITY	IN	NUMBER			x
FIRM_PLANNED_TYPE	IN	NUMBER	x		
FIRM_QUANTITY	IN	NUMBER			x
FIRM_DATE	IN	DATE			x
IMPLEMENT_DEMAND_CLASS	IN	VARCHAR2(34)			x
IMPLEMENT_DATE	IN	DATE			x
IMPLEMENT_QUANTITY	IN	NUMBER			x
IMPLEMENT_FIRM	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
IMPLEMENT_WIP_CLASS_CODE	IN	VARCHAR2(10)			x
IMPLEMENT_JOB_NAME	IN	VARCHAR2(240)			x
IMPLEMENT_DOCK_DATE	IN	DATE			x
IMPLEMENT_STATUS_CODE	IN	NUMBER			x
IMPLEMENT_UOM_CODE	IN	VARCHAR2(3)			x
IMPLEMENT_LOCATION_ID	IN	NUMBER			x
IMPLEMENT_SOURCE_ORG_ID	IN	NUMBER			x
IMPLEMENT_SUPPLIER_ID	IN	NUMBER			x
IMPLEMENT_SUPPLIER_SITE_ID	IN	NUMBER			x
IMPLEMENT_AS	IN	NUMBER			x
RELEASE_STATUS	IN	NUMBER			x
LOAD_TYPE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
PROCESS_SEQ_ID	IN	NUMBER			x
SCO_SUPPLY_FLAG	IN	NUMBER			x
ALTERNATE_BOM_DESIGNATOR	IN	VARCHAR2(10)			x
ALTERNATE_ROUTING_DESIGNATOR	IN	VARCHAR2(10)			x
OPERATION_SEQ_NUM	IN	NUMBER			x
SOURCE	IN	NUMBER			x
BY_PRODUCT_USING_ASSY_ID	IN	NUMBER			x
SOURCE_ORGANIZATION_ID	IN	NUMBER			x
SOURCE_SR_INSTANCE_ID	IN	NUMBER			x
SOURCE_SUPPLIER_SITE_ID	IN	NUMBER			x
SOURCE_SUPPLIER_ID	IN	NUMBER			x
SHIP_METHOD	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
WEIGHT_CAPACITY_USED	IN	NUMBER			x
VOLUME_CAPACITY_USED	IN	NUMBER			x
SOURCE_SUPPLY_SCHEDULE_NAME	IN	NUMBER			x
NEW_SHIP_DATE	IN	DATE			x
NEW_DOCK_DATE	IN	DATE			x
LINE_ID	IN	NUMBER			x
PROJECT_ID	IN	NUMBER(15)			x
TASK_ID	IN	NUMBER(15)			x
PLANNING_GROUP	IN	VARCHAR2(30)			x
IMPLEMENT_PROJECT_ID	IN	NUMBER(15)			x
IMPLEMENT_TASK_ID	IN	NUMBER(15)			x
IMPLEMENT_SCHEDULE_GROUP_ID	IN	NUMBER			x
IMPLEMENT_BUILD_SEQUENCE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
IMPLEMENT_ALTERNATE_BOM	IN	VARCHAR2(10)			x
IMPLEMENT_ALTERNATE_ROUTING	IN	VARCHAR2(10)			x
IMPLEMENT_UNIT_NUMBER	IN	VARCHAR2(30)			x
IMPLEMENT_LINE_ID	IN	NUMBER			x
RELEASE_ERRORS	IN	VARCHAR2(1)			x
NUMBER1	IN	NUMBER			x
SOURCE_ITEM_ID	IN	NUMBER			x
ORDER_NUMBER	IN	VARCHAR2(240)			x
SCHEDULE_GROUP_ID	IN	NUMBER			x
SCHEDULE_GROUP_NAME	IN	VARCHAR2(30)			x
BUILD_SEQUENCE	IN	NUMBER			x
WIP_ENTITY_ID	IN	NUMBER			x
WIP_ENTITY_NAME	IN	VARCHAR2(240)			x

Parameter	Usage	Type	Required	Derived	Optional
WO_LATENESS_COST	IN	NUMBER			x
IMPLEMENT_PROCESSING_DAYS	IN	NUMBER			x
DELIVERY_PRICE	IN	NUMBER			x
LATE_SUPPLY_DATE	IN	DATE			x
LATE_SUPPLY_QTY	IN	NUMBER			x
SUBINVENTORY_CODE	IN	VARCHAR2(10)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
SCHEDULE_DESIGNATOR	IN	VARCHAR2(10)			x
VENDOR_ID	IN	NUMBER			x
VENDOR_SITE_ID	IN	NUMBER			x
SUPPLIER_SITE_ID	IN	NUMBER			x
PURCH_ORDER_ID	IN	NUMBER			x
EXPECTED_SCRAP_QTY	IN	NUMBER			x
QTY_SCRAPPED	IN	NUMBER			x
QTY_COMPLETED	IN	NUMBER			x
LOT_NUMBER	IN	VARCHAR2(30)			x
EXPIRATION_DATE	IN	DATE			x
WIP_STATUS_CODE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
DAILY_RATE	IN	NUMBER			x
LOCATOR_ID	IN	NUMBER			x
SERIAL_NUMBER	IN	VARCHAR2(30)			x
REFRESH_ID	IN	NUMBER			x
LOCATOR_NAME	IN	VARCHAR2(204)			x
ONHAND_SOURCE_TYPE	IN	NUMBER			x
SR_MTL_SUPPLY_ID	IN	NUMBER			x
DEMAND_CLASS	IN	VARCHAR2(34)			
FROM_ORGANIZATION_ID	IN	NUMBER			
WIP_SUPPLY_TYPE	IN	NUMBER			
PO_LINE_ID	IN	NUMBER			

PLAN_ID

Plan identifier

TRANSACTION_ID

Transaction unique identifier

INVENTORY_ITEM_ID

Inventory item identifier

ORGANIZATION_ID

Organization identifier

SCHEDULE_DESIGNATOR_ID

Schedule designator identifier

SOURCE_SCHEDULE_NAME

Source schedule name

REVISION

Inventory item revision code

UNIT_NUMBER

Unit number

NEW_SCHEDULE_DATE

End date of the supply (completion date of first unit)

OLD_SCHEDULE_DATE

Old schedule date

NEW_WIP_START_DATE

New WIP schedule start date

OLD_WIP_START_DATE

Old WIP schedule start date

FIRST_UNIT_COMPLETION_DATE

First unit completion date for recommended repetitive schedules

LAST_UNIT_COMPLETION_DATE

Last unit completion date for recommended repetitive schedules

FIRST_UNIT_START_DATE

First unit start date for repetitive schedule

LAST_UNIT_START_DATE

Last unit start date for repetitive schedule

DISPOSITION_ID

Identifier which references to source of supply

DISPOSITION_STATUS_TYPE

Disposition type code

ORDER_TYPE

Specifies type of order: planned order, purchase order, etc...

SUPPLIER_ID

Supplier identifier

NEW_ORDER_QUANTITY

Supply quantity

OLD_ORDER_QUANTITY

Old order quantity

NEW_ORDER_PLACEMENT_DATE

New order placement date

OLD_ORDER_PLACEMENT_DATE

Old order placement date

RESCHEDULE_DAYS

Different between old and new schedule dates

RESCHEDULE_FLAG

Flag indicating if this row been rescheduled

SCHEDULE_COMPRESS_DAYS

Schedule compress days

NEW_PROCESSING_DAYS

Repetitive schedule processing days

PURCH_LINE_NUM

Purchase order line number (for purchase order)

QUANTITY_IN_PROCESS

Quantity being processed by the WIP/PO interface processes

IMPLEMENTED_QUANTITY

Planned order implemented quantity

FIRM_PLANNED_TYPE

Flag indicating whether the order is firm

FIRM_QUANTITY

Firm quantity

FIRM_DATE

Firm date

IMPLEMENT_DEMAND_CLASS

Implement demand class

IMPLEMENT_DATE

Implement due date

IMPLEMENT_QUANTITY

Planned order implemented quantity

IMPLEMENT_FIRM

Implement firm flag

IMPLEMENT_WIP_CLASS_CODE

Implement WIP class code

IMPLEMENT_JOB_NAME

Implement job name

IMPLEMENT_DOCK_DATE

Implement dock date

IMPLEMENT_STATUS_CODE

Implement status code

IMPLEMENT_UOM_CODE

Implement unit of measure code

IMPLEMENT_LOCATION_ID

Implement location identifier

IMPLEMENT_SOURCE_ORG_ID

Implement source organization identifier

IMPLEMENT_SUPPLIER_ID

Implement supplier identifier

IMPLEMENT_SUPPLIER_SITE_ID

Implement supplier site identifier

IMPLEMENT_AS

Implement order type

RELEASE_STATUS

Release status code

LOAD_TYPE

Load program to execute

PROCESS_SEQ_ID

Process sequence identifier

SCO_SUPPLY_FLAG

Flag to indicate if supply was suggested by SCO

ALTERNATE_BOM_DESIGNATOR

Alternate BOM designator

ALTERNATE_ROUTING_DESIGNATOR

Alternate routing designator

OPERATION_SEQ_NUM

Operation sequence number

SOURCE

BY_PRODUCT_USING_ASSY_ID

SOURCE_ORGANIZATION_ID

Source organization identifier

SOURCE_SR_INSTANCE_ID

Source org instance identifier

SOURCE_SUPPLIER_SITE_ID

Source supplier site identifier

SOURCE_SUPPLIER_ID

Source supplier identifier

SHIP_METHOD

Ship method

WEIGHT_CAPACITY_USED

Weight capacity used

VOLUME_CAPACITY_USED

Volume capacity used

SOURCE_SUPPLY_SCHEDULE_NAME

Source supply schedule name

NEW_SHIP_DATE

New ship date

NEW_DOCK_DATE

New suggested dock date

LINE_ID

Manufacturing line identifier

PROJECT_ID

Project identifier

TASK_ID

Task identifier

PLANNING_GROUP

Planning group code

IMPLEMENT_PROJECT_ID

Implement project identifier

IMPLEMENT_TASK_ID

Implement task identifier

IMPLEMENT_SCHEDULE_GROUP_ID

Implement schedule group identifier

IMPLEMENT_BUILD_SEQUENCE

Implement build sequence for the planned order to be implemented as a discrete job

IMPLEMENT_ALTERNATE_BOM

Implement alternate BOM designator

IMPLEMENT_ALTERNATE_ROUTING

Implement alternate routing

IMPLEMENT_UNIT_NUMBER

Implement unit number

IMPLEMENT_LINE_ID

Implement line identifier

RELEASE_ERRORS

NUMBER1

SOURCE_ITEM_ID

Source item identifier

ORDER_NUMBER

Order number

SCHEDULE_GROUP_ID

Schedule group identifier

SCHEDULE_GROUP_NAME

Schedule group name

BUILD_SEQUENCE

Build Sequence for the Planned Order

WIP_ENTITY_ID

WIP entity identifier

WIP_ENTITY_NAME

WIP entity name

WO_LATENESS_COST

Work order lateness cost

IMPLEMENT_PROCESSING_DAYS

Implement processing days

DELIVERY_PRICE

Supply unit price for purchasing supply

LATE_SUPPLY_DATE

Supply date for the shadow part of the split supplies

LATE_SUPPLY_QTY

Shadow supply quantity

SUBINVENTORY_CODE

Sub-inventory code

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

SCHEDULE_DESIGNATOR

Schedule designator

VENDOR_ID

Supplier identifier

VENDOR_SITE_ID

Supplier site identifier

SUPPLIER_SITE_ID

Supplier site identifier

PURCH_ORDER_ID

Purchase order identifier

EXPECTED_SCRAP_QTY

Expected scrap qty

QTY_SCRAPPED

Current job scrapped units

QTY_COMPLETED

Current job quantity completed

LOT_NUMBER

Lot number for on-hand quantities

EXPIRATION_DATE

Expiration date

WIP_STATUS_CODE

WIP job status code

DAILY_RATE

Daily rate for recommended repetitive schedules

LOCATOR_ID

Locator identifier

SERIAL_NUMBER

Serial number

REFRESH_ID

Refresh identifier

LOCATOR_NAME

Locator name

ONHAND_SOURCE_TYPE

Onhand source type

SR_MTL_SUPPLY_ID

Supply identifier from the source

DEMAND_CLASS

Demand class code

FROM_ORGANIZATION_ID

From organization identifier

WIP_SUPPLY_TYPE

WIP supply type

PO_LINE_ID

Purchase order line identifier

MSC_ST_SYSTEM_ITEMS

The staging table used by the collection program to validate and process data for table MSC_SYSTEM_ITEMS.

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_ID	IN	NUMBER	x		
SR_ORGANIZATION_ID	IN	NUMBER			x
INVENTORY_ITEM_ID	IN	NUMBER			x
SR_INVENTORY_ITEM_ID	IN	NUMBER	x		
ITEM_NAME	IN	VARCHAR2(40)			x
LOTS_EXPIRATION	IN	NUMBER			x
LOT_CONTROL_CODE	IN	NUMBER	x		
SHRINKAGE_RATE	IN	NUMBER			x
FIXED_DAYS_SUPPLY	IN	NUMBER			x
FIXED_ORDER_QUANTITY	IN	NUMBER			x
FIXED_LOT_MULTIPLIER	IN	NUMBER			x
MINIMUM_ORDER_QUANTITY	IN	NUMBER			x
MAXIMUM_ORDER_QUANTITY	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
ROUNDING_CONTROL_TYPE	IN	NUMBER	x		
PLANNING_TIME_FENCE_DAYS	IN	NUMBER			x
DEMAND_TIME_FENCE_DAYS	IN	NUMBER			x
RELEASE_TIME_FENCE_CODE	IN	NUMBER			x
RELEASE_TIME_FENCE_DAYS	IN	NUMBER			x
DESCRIPTION	IN	VARCHAR2(240)			x
IN_SOURCE_PLAN	IN	NUMBER	x		
REVISION	IN	VARCHAR2(3)			x
SR_CATEGORY_ID	IN	NUMBER			x
CATEGORY_NAME	IN	VARCHAR2(200)			x
ABC_CLASS_ID	IN	NUMBER			x
ABC_CLASS_NAME	IN	VARCHAR2(40)			x

Parameter	Usage	Type	Required	Derived	Optional
MRP_PLANNING_CODE	IN	NUMBER	x		
FIXED_LEAD_TIME	IN	NUMBER			x
VARIABLE_LEAD_TIME	IN	NUMBER			x
PREPROCESSING_LEAD_TIME	IN	NUMBER			x
POSTPROCESSING_LEAD_TIME	IN	NUMBER			x
FULL_LEAD_TIME	IN	NUMBER	x		
CUMULATIVE_TOTAL_LEAD_TIME	IN	NUMBER			x
CUM_MANUFACTURING_LEAD_TIME	IN	NUMBER			x
UOM_CODE	IN	VARCHAR2(3)	x		
UNIT_WEIGHT	IN	NUMBER			x
UNIT_VOLUME	IN	NUMBER			x
WEIGHT_UOM	IN	VARCHAR2(3)			x

Parameter	Usage	Type	Required	Derived	Optional
VOLUME_UOM	IN	VARCHAR2(3)			x
PRODUCT_FAMILY_ID	IN	NUMBER			x
ATP_RULE_ID	IN	NUMBER			x
MRP_CALCULATE_ATP_FLAG	IN	NUMBER	x		
ATP_COMPONENTS_FLAG	IN	VARCHAR2(1)	x		
BUILT_IN_WIP_FLAG	IN	NUMBER	x		
PURCHASING_ENABLED_FLAG	IN	NUMBER	x		
PLANNING_MAKE_BUY_CODE	IN	NUMBER	x		
REPETITIVE_TYPE	IN	NUMBER	x		
STANDARD_COST	IN	NUMBER			x
CARRYING_COST	IN	NUMBER			x
ORDER_COST	IN	NUMBER			x
DMD_LATENESS_COST	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
SS_PENALTY_COST	IN	NUMBER			x
SUPPLIER_CAP_OVERUTIL_COST	IN	NUMBER			x
LIST_PRICE	IN	NUMBER			x
AVERAGE_DISCOUNT	IN	NUMBER			x
END_ASSEMBLY_PEGGING_FLAG	IN	VARCHAR2(1)			x
END_ASSEMBLY_PEGGING	IN	NUMBER			x
FULL_PEGGING	IN	NUMBER	x		
ENGINEERING_ITEM_FLAG	IN	NUMBER	x		
WIP_SUPPLY_TYPE	IN	NUMBER	x		
MRP_SAFETY_STOCK_CODE	IN	NUMBER			x
MRP_SAFETY_STOCK_PERCENT	IN	NUMBER			x
SAFETY_STOCK_BUCKET_DAYS	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_USE_UP_DATE	IN	DATE			x
BUYER_NAME	IN	VARCHAR2(240)			x
PLANNER_CODE	IN	VARCHAR2(10)			x
PLANNING_EXCEPTION_SET	IN	VARCHAR2(10)			x
EXCESS_QUANTITY	IN	NUMBER			x
EXCEPTION_SHORTAGE_DAYS	IN	NUMBER			x
EXCEPTION_EXCESS_DAYS	IN	NUMBER			x
EXCEPTION_OVERPROMISED_DAYS	IN	NUMBER			x
REPETITIVE_VARIANCE_DAYS	IN	NUMBER			x
BASE_ITEM_ID	IN	NUMBER			x
BOM_ITEM_TYPE	IN	NUMBER			x
ATO_FORECAST_CONTROL	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
ORGANIZATION_CODE	IN	VARCHAR2(7)			x
EFFECTIVITY_CONTROL	IN	NUMBER	x		
ACCEPTABLE_EARLY_DELIVERY	IN	NUMBER			x
INVENTORY_PLANNING_CODE	IN	NUMBER	x		
INVENTORY_TYPE	IN	NUMBER			x
ACCEPTABLE_RATE_INCREASE	IN	NUMBER			x
ACCEPTABLE_RATE_DECREASE	IN	NUMBER			x
PRIMARY_SUPPLIER_ID	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
ATP_FLAG	IN	VARCHAR2(1)	x		
INVENTORY_ITEM_FLAG	IN	NUMBER			x
REVISION_QTY_CONTROL_CODE	IN	NUMBER			x
EXPENSE_ACCOUNT	IN	NUMBER			x
INVENTORY_ASSET_FLAG	IN	VARCHAR2(1)			x
BUYER_ID	IN	NUMBER(9)			x
MATERIAL_COST	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
RESOURCE_COST	IN	NUMBER			x
SOURCE_ORG_ID	IN	NUMBER			x
PICK_COMPONENTS_FLAG	IN	VARCHAR2(1)			x

ORGANIZATION_ID

Organization identifier

SR_ORGANIZATION_ID

Source organization identifier

INVENTORY_ITEM_ID

Inventory item identifier

SR_INVENTORY_ITEM_ID

Source inventory item identifier

ITEM_NAME

Item name

LOTS_EXPIRATION

Lots expiration

LOT_CONTROL_CODE

Flag indicating if lots_expiration is used or not

SHRINKAGE_RATE

Percentage of shrinkage for this item

FIXED_DAYS_SUPPLY

Period of the supply days

FIXED_ORDER_QUANTITY

Fixed order quantity

FIXED_LOT_MULTIPLIER

Fixed lot multiplier

MINIMUM_ORDER_QUANTITY

Minimum size of an order

MAXIMUM_ORDER_QUANTITY

Maximum size of an order

ROUNDING_CONTROL_TYPE

Flag indicating if rounding of the quantity is allowed

PLANNING_TIME_FENCE_DAYS

Planning time fences days of the item

DEMAND_TIME_FENCE_DAYS

Demand time fence days

RELEASE_TIME_FENCE_CODE

Release time fence code

RELEASE_TIME_FENCE_DAYS

Release time fence days

DESCRIPTION

Item description

IN_SOURCE_PLAN

Flag indicating whether the item is in the plan

REVISION

Item revision code

SR_CATEGORY_ID

Source category identifier

CATEGORY_NAME

Category name

ABC_CLASS_ID

ABC class identifier

ABC_CLASS_NAME

ABC class name

MRP_PLANNING_CODE

MRP planning code

FIXED_LEAD_TIME

Fixed lead time

VARIABLE_LEAD_TIME

Variable lead time

PREPROCESSING_LEAD_TIME

Preprocessing lead time

POSTPROCESSING_LEAD_TIME

Postprocessing lead time

FULL_LEAD_TIME

Full lead time

CUMULATIVE_TOTAL_LEAD_TIME

Cumulative total lead time

CUM_MANUFACTURING_LEAD_TIME

Cumulative manufacturing lead time

UOM_CODE

Unit of measure code

UNIT_WEIGHT

Weight of the item

UNIT_VOLUME

Volume of the item

WEIGHT_UOM

Unit of measure for the weight

VOLUME_UOM

Unit of measure for the volume

PRODUCT_FAMILY_ID

Product family identifier

ATP_RULE_ID

ATP rule identifier

MRP_CALCULATE_ATP_FLAG

Flag indication whether to calculate ATP in MRP

ATP_COMPONENTS_FLAG

Flag indicating whether to calculate components ATP

BUILT_IN_WIP_FLAG

Flag to indicate if the item can be built in WIP

PURCHASING_ENABLED_FLAG

Flag to indicate if the item can be purchased

PLANNING_MAKE_BUY_CODE

Plan this item as either a make item or buy item

REPETITIVE_TYPE

Flag indicates if this item build repetitively

STANDARD_COST

Standard cost

CARRYING_COST

Actual carrying cost

ORDER_COST

Order cost

DMD_LATENESS_COST

DMD lateness cost

SS_PENALTY_COST

SS penalty cost

SUPPLIER_CAP_OVERUTIL_COST

Supplier capacity over-utilization cost

LIST_PRICE

Item list price

AVERAGE_DISCOUNT

Item average discount

END_ASSEMBLY_PEGGING_FLAG

Peg to the end assembly on reports

END_ASSEMBLY_PEGGING

Peg to the end assembly on reports (value is populated by the plan)

FULL_PEGGING

Full pegging flag

ENGINEERING_ITEM_FLAG

Engineering item flag

WIP_SUPPLY_TYPE

WIP supply type

MRP_SAFETY_STOCK_CODE

Safety stock code

MRP_SAFETY_STOCK_PERCENT

Safety stock percent

SAFETY_STOCK_BUCKET_DAYS

Safety stock bucket days

INVENTORY_USE_UP_DATE

Use up date

BUYER_NAME

Buyer name

PLANNER_CODE

Planner code

PLANNING_EXCEPTION_SET

Exception control set

EXCESS_QUANTITY

Excess quantity

EXCEPTION_SHORTAGE_DAYS

Exception shortage days

EXCEPTION_EXCESS_DAYS

Exception excess days

EXCEPTION_OVERPROMISED_DAYS

Exception overpromised days

REPETITIVE_VARIANCE_DAYS

Repetitive variance days

BASE_ITEM_ID

Inventory base item identifier

BOM_ITEM_TYPE

BOM item type

ATO_FORECAST_CONTROL

ATO forecast control

ORGANIZATION_CODE

Organization code

EFFECTIVITY_CONTROL

Effectivity control code

ACCEPTABLE_EARLY_DELIVERY

Acceptable early delivery

INVENTORY_PLANNING_CODE

Inventory planning code

INVENTORY_TYPE

Inventory type

ACCEPTABLE_RATE_INCREASE

Acceptable rate increase

ACCEPTABLE_RATE_DECREASE

Acceptable rate increase

PRIMARY_SUPPLIER_ID

Primary supplier identifier

DELETED_FLAG

Peg to the end assembly on reports

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

ATP_FLAG

ATP flag

INVENTORY_ITEM_FLAG

Inventory item identifier

REVISION_QTY_CONTROL_CODE

Revision quantity control

EXPENSE_ACCOUNT

Expense account

INVENTORY_ASSET_FLAG

Inventory asset flag

BUYER_ID

Buyer identifier

MATERIAL_COST

Material cost

RESOURCE_COST

Resource cost

SOURCE_ORG_ID

Source organization identifier

PICK_COMPONENTS_FLAG

Flag indicating whether all shippable components should be picked

MSC_ST_TRADING_PARTNERS

The staging table used by the collection program to validate and process data for table MSC_TRADING_PARTNERS.

Parameter	Usage	Type	Required	Derived	Optional
PARTNER_ID	IN	NUMBER			x
ORGANIZATION_CODE	IN	VARCHAR2(7)			x
SR_TP_ID	IN	NUMBER	x		
DISABLE_DATE	IN	DATE			x
STATUS	IN	VARCHAR2(1)			x
MASTER_ORGANIZATION	IN	NUMBER			x
PARTNER_TYPE	IN	NUMBER	x		
PARTNER_NAME	IN	VARCHAR2(80)			x
PARTNER_NUMBER	IN	VARCHAR2(154)			x
CALENDAR_CODE	IN	VARCHAR2(14)			x
CALENDAR_EXCEPTION_SET_ID	IN	NUMBER			x
OPERATING_UNIT	IN	NUMBER			x
MAXIMUM_WEIGHT	IN	NUMBER			x
MAXIMUM_VOLUME	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
WEIGHT_UOM	IN	VARCHAR2(3)			x
VOLUME_UOM	IN	VARCHAR2(3)			x
PROJECT_REFERENCE_ENABLED	IN	NUMBER			x
PROJECT_CONTROL_LEVEL	IN	NUMBER			x
DEMAND_LATENESS_COST	IN	NUMBER			x
SUPPLIER_CAP_OVERUTIL_COST	IN	NUMBER			x
RESOURCE_CAP_OVERUTIL_COST	IN	NUMBER			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
MODELED_CUSTOMER_ID	IN	NUMBER			x
MODELED_CUSTOMER_SITE_ID	IN	NUMBER			x
MODELED_SUPPLIER_ID	IN	NUMBER			x
MODELED_SUPPLIER_SITE_ID	IN	NUMBER			x
TRANSPORT_CAP_OVER_UTIL_COST	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
USE_PHANTOM_ROUTINGS	IN	NUMBER			x
INHERIT_PHANTOM_OP_SEQ	IN	NUMBER			x
DEFAULT_ATP_RULE_ID	IN	NUMBER			x
DEFAULT_DEMAND_CLASS	IN	VARCHAR2(34)			x
MATERIAL_ACCOUNT	IN	NUMBER			x
EXPENSE_ACCOUNT	IN	NUMBER			x
SOURCE_ORG_ID	IN	NUMBER			x
ORGANIZATION_TYPE	IN	NUMBER			x

PARTNER_ID

Unique partner identifier which can be customer id, supplier id, or inventory organization id

ORGANIZATION_CODE

Organization code

SR_TP_ID

Unique partner identifier in the source application instance

DISABLE_DATE

Disable date of the trading partner

STATUS

Status of the trading partner

MASTER_ORGANIZATION

Master organization identifier

PARTNER_TYPE

Specify the type of partner: Customer, Supplier, or organization

PARTNER_NAME

Name of the supplier or customer

PARTNER_NUMBER

Number of the supplier or customer

CALENDAR_CODE

Calendar used for this partner. The code includes instance code and calendar code from the source apps.

CALENDAR_EXCEPTION_SET_ID

Calendar exception set identifier

OPERATING_UNIT

Operating unit

MAXIMUM_WEIGHT

Maximum weight

MAXIMUM_VOLUME

Maximum volume

WEIGHT_UOM

Weight unit of measure

VOLUME_UOM

Volume unit of measure

PROJECT_REFERENCE_ENABLED

Project reference enabled flag

PROJECT_CONTROL_LEVEL

Project control level

DEMAND_LATENESS_COST

Demand lateness cost

SUPPLIER_CAP_OVERUTIL_COST

Supplier over-utilization cost

RESOURCE_CAP_OVERUTIL_COST

Resource over-utilization cost

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MODELED_CUSTOMER_ID

Customer identifier which is modeled as inventory organization

MODELED_CUSTOMER_SITE_ID

Customer site identifier which is modeled as inventory organization

MODELED_SUPPLIER_ID

Supplier identifier which is modeled as inventory organization

MODELED_SUPPLIER_SITE_ID

Supplier site identifier which is modeled as inventory organization

TRANSPORT_CAP_OVER_UTIL_COST

Transportation over-utilization cost

USE_PHANTOM_ROUTINGS

Use phantom routings

INHERIT_PHANTOM_OP_SEQ

Inherit phantom op sequence

DEFAULT_ATP_RULE_ID

Default ATP rule identifier

DEFAULT_DEMAND_CLASS

Default demand class

MATERIAL_ACCOUNT

Material account

EXPENSE_ACCOUNT

Expense account

SOURCE_ORG_ID

Organization to source items from

ORGANIZATION_TYPE

Organization

MSC_ST_TRADING_PARTNER_SITES

The staging table used by the collection program to validate and process data for table MSC_TRADING_PARTNER_SITES.

Parameter	Usage	Type	Required	Derived	Optional
PARTNER_ID	IN	NUMBER			x
PARTNER_SITE_ID	IN	NUMBER			x
PARTNER_ADDRESS	IN	VARCHAR2(1600)			x
SR_TP_ID	IN	NUMBER(15)	x		
SR_TP_SITE_ID	IN	NUMBER	x		
TP_SITE_CODE	IN	VARCHAR2(30)			x
LOCATION	IN	VARCHAR2(40)			x
PARTNER_TYPE	IN	NUMBER	x		
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x
LONGITUDE	IN	NUMBER(10, 7)			x
LATITUDE	IN	NUMBER(10, 7)			x
OPERATING_UNIT_NAME	IN	VARCHAR2(60)			x

PARTNER_ID

Trading partner unique identifier

PARTNER_SITE_ID

Trading partner site unique identifier

PARTNER_ADDRESS

Trading partner address

SR_TP_ID

Trading partner unique identifier from source application

SR_TP_SITE_ID

Trading partner site unique identifier from source application

TP_SITE_CODE

Site code

LOCATION

Partner location

PARTNER_TYPE

Indicate type of partner: Customer, Supplier, or Organization

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

LONGITUDE

Longitude

LATITUDE

Latitude

OPERATING_UNIT_NAME

MSC_ST_UNITS_OF_MEASURE

The staging table used by the collection program to validate and process data for table MSC_UNITS_OF_MEASURE.

Parameter	Usage	Type	Required	Derived	Optional
UNIT_OF_MEASURE	IN	VARCHAR2(25)	x		
UOM_CODE	IN	VARCHAR2(3)	x		
UOM_CLASS	IN	VARCHAR2(10)	x		
BASE_UOM_FLAG	IN	VARCHAR2(1)	x		

Parameter	Usage	Type	Required	Derived	Optional
DISABLE_DATE	IN	DATE			x
DESCRIPTION	IN	VARCHAR2(50)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

UNIT_OF_MEASURE

Unit of measure name

UOM_CODE

Abbreviated unit of measure code

UOM_CLASS

Unit of measure class

BASE_UOM_FLAG

Base unit of measure flag

DISABLE_DATE

Date when the unit can no longer be used to define conversions

DESCRIPTION

Unit of measure description

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_UNIT_NUMBERS

The staging table used by the collection program to validate and process data for table MSC_UNIT_NUMBERS.

Parameter	Usage	Type	Required	Derived	Optional
UNIT_NUMBER	IN	VARCHAR2(30)	x		
END_ITEM_ID	IN	NUMBER	x		
MASTER_ORGANIZATION_ID	IN	NUMBER	x		
COMMENTS	IN	VARCHAR2(240)			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

UNIT_NUMBER

Unit number

END_ITEM_ID

End item unique identifier

MASTER_ORGANIZATION_ID

Master organization identifier

COMMENTS

Comments

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_UOM_CLASS_CONVERSIONS

The staging table used by the collection program to validate and process data for table MSC_UOM_CLASS_CONVERSIONS.

Parameter	Usage	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	IN	NUMBER	x		
FROM_UNIT_OF_MEASURE	IN	VARCHAR2(25)	x		
FROM_UOM_CODE	IN	VARCHAR2(3)	x		
FROM_UOM_CLASS	IN	VARCHAR2(10)	x		
TO_UNIT_OF_MEASURE	IN	VARCHAR2(25)	x		
TO_UOM_CODE	IN	VARCHAR2(3)	x		
TO_UOM_CLASS	IN	VARCHAR2(10)	x		
CONVERSION_RATE	IN	NUMBER	x		
DISABLE_DATE	IN	DATE			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	
CREATED_BY	IN	NUMBER		x	

Parameter	Usage	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

INVENTORY_ITEM_ID

The inventory item for which the conversion factors between base units of measure

FROM_UNIT_OF_MEASURE

Base unit of measure of the items base class

FROM_UOM_CODE

Base unit of measure short name for the items base class

FROM_UOM_CLASS

Base class of the item

TO_UNIT_OF_MEASURE

Base unit of the class to which the conversion is defined

TO_UOM_CODE

Base unit short name of the class to which the conversion is defined

TO_UOM_CLASS

Class to which the conversion is defined

CONVERSION_RATE

Conversion rate from the items class base unit to the "to" class base unit

DISABLE_DATE

Date when the defined inter-class conversion can no longer be used

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

MSC_ST_UOM_CONVERSIONS

The staging table used by the collection program to validate and process data for table MSC_UOM_CONVERSIONS.

Parameter	Usage	Type	Required	Derived	Optional
UNIT_OF_MEASURE	IN	VARCHAR2(25)	x		
UOM_CODE	IN	VARCHAR2(3)	x		
UOM_CLASS	IN	VARCHAR2(10)	x		
INVENTORY_ITEM_ID	IN	NUMBER	x		
CONVERSION_RATE	IN	NUMBER	x		
DEFAULT_CONVERSION_FLAG	IN	VARCHAR2(1)	x		
DISABLE_DATE	IN	DATE			x
DELETED_FLAG	IN	NUMBER	x		
LAST_UPDATE_DATE	IN	DATE		x	
LAST_UPDATED_BY	IN	NUMBER		x	
CREATION_DATE	IN	DATE		x	

Parameter	Usage	Type	Required	Derived	Optional
CREATED_BY	IN	NUMBER		x	
LAST_UPDATE_LOGIN	IN	NUMBER		x	
REQUEST_ID	IN	NUMBER		x	
PROGRAM_APPLICATION_ID	IN	NUMBER		x	
PROGRAM_ID	IN	NUMBER		x	
PROGRAM_UPDATE_DATE	IN	DATE		x	
SR_INSTANCE_ID	IN	NUMBER	x		
REFRESH_ID	IN	NUMBER			x

UNIT_OF_MEASURE

Primary unit of measure long name

UOM_CODE

Unit of measure code

UOM_CLASS

Destination class of conversion

INVENTORY_ITEM_ID

Inventory item identifier

CONVERSION_RATE

Conversion rate from conversion unit to base unit of class

DEFAULT_CONVERSION_FLAG

Indicates whether the conversion factor applies for this item or it is defined as standard conversion factor

DISABLE_DATE

Date when the conversion is no longer valid to be used in the system (transactions, etc.)

DELETED_FLAG

Yes/No flag indicates whether corresponding record in ODS will be deleted

LAST_UPDATE_DATE

Standard Who column

LAST_UPDATED_BY

Standard Who column

CREATION_DATE

Standard Who column

CREATED_BY

Standard Who column

LAST_UPDATE_LOGIN

Standard Who column

REQUEST_ID

Concurrent Who column

PROGRAM_APPLICATION_ID

Concurrent Who column

PROGRAM_ID

Concurrent Who column

PROGRAM_UPDATE_DATE

Concurrent Who column

SR_INSTANCE_ID

Source application instance identifier

REFRESH_ID

Refresh identifier

5

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Open Interfaces and APIs

This chapter covers the following topics:

- Open Forecast Interface
- Validation
- Resolving Failed Open Forecast Interface Rows
- Open Master Schedule Interface
- Validation
- Resolving Failed Open Master Schedule Interface Rows
- Open Forecast Entries Application Program Interface
- Open Forecast Interface Designator Table Description
- Validation
- Using the Open Forecast Entries API
- Sourcing Rule Application Program Interface
- Setting Up the Sourcing Rule/Bill of Distribution API
- Record Parameter Descriptions
- Validation of Sourcing Rule /Bill of Distribution API
- Sourcing Rule Assignment API Features
- Setting Up the Sourcing Rule Assignment API
- Record Parameter Descriptions
- Validation of Sourcing Rule Assignment API

Open Forecast Interface

You can import forecasts from any source using the Open Forecast Interface table. Oracle Master Scheduling/MRP automatically validates and implements imported forecasts as new forecasts in Oracle Master Scheduling/MRP.

The purpose of this essay is to explain how to use the Open Forecast Interface so that you can integrate other applications with Oracle Master Scheduling/MRP.

Functional Overview

All processing is performed by the Forecast Interface Load program. The Forecast Interface Load program is launched by the Planning Manager, which periodically checks the Open Forecast Interface to see if there are any new rows waiting to be processed.

Setting Up the Open Forecast Interface

You must define at least one organization, item, forecast set, and forecast name before using the Open Forecast Interface. Since the Planning Manager decides when to call the Forecast Interface Load program, the Planning Manager must also be running before you can import forecasts via the Open Forecast Interface.

Inserting into the Open Forecast Interface Table

You must load your forecasts into the MRP_FORECAST_INTERFACE table. The Forecast Interface Load program validates your forecasts, derives any additional data as necessary, and then processes it by creating new forecasts in Oracle Master Scheduling/MRP.

Open Forecast Interface Table Description

The Open Forecast Interface Table is described in the following table. This is typically used for batch loads, performed when the load is low on the concurrent processing system. It is not interactive.

Column Name	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	Number			
FORECAST_DESIGNATOR	Varchar2(10)	x		
ORGANIZATION_ID	Number	x		

Column Name	Type	Required	Derived	Optional
FORECAST_DATE	Date	x		
LAST_UPDATE_DATE	Date	x		
LAST_UPDATED_BY	Number	x		
CREATION_DATE	Date	x		
CREATED_BY	Number	x		
LAST_UPDATE_LOGIN	Number			x
QUANTITY	Number	x		
PROCESS_STATUS	Number	x		
CONFIDENCE_PERCENTAGE	Number	x		
COMMENTS	Varchar2(240)			x
ERROR_MESSAGE	Varchar2(240)		x	
REQUEST_ID	Number		x	
PROGRAM_APPLICATION_ID	Number		x	
PROGRAM_ID	Number		x	
PROGRAM_UPDATE_DATE	Date		x	

Column Name	Type	Required	Derived	Optional	
WORKDAY_CONTROL	Number			x	
BUCKET_TYPE	Number			x	
FORECAST_END_DATE	Date				**x**
TRANSACTION_ID	Number			x	
SOURCE_CODE	Varchar2(10)			x	
SOURCE_LINE_ID	Number			x	
ATTRIBUTE_CATEGORY	Varchar230)			x	
ATTRIBUTE1 - ATTRIBUTE15	Varchar2(150)			x	
PROJECT ID	Number(15)			x	
TASK ID	Number(15			x	
LINE ID	Number(15			x	

Legend

1: Multiple Period Forecast Entries only

Required Data

ORGANIZATION_ID, FORECAST_DESIGNATOR, INVENTORY_ITEM_ID, FORECAST_DATE, and QUANTITY are used by the Forecast Interface Load program to create new forecast entries in MRP_FORECAST_DATES.

PROCESS_STATUS indicates the current state of processing of each new forecast entry in the Open Forecast Interface. Valid values include:

- 1. Do not process

- 2. Waiting to be processed
- 3. Being processed
- 4. Error
- 5. Processed

When you first load a new forecast entry into the Open Forecast Interface, set PROCESS_STATUS to 2 (Waiting to be processed).

Derived Data

ERROR_MESSAGE indicates a problem that prevents the Forecast Interface Load program from successfully processing a new forecast entry in the Open Forecast Interface.

The Forecast Interface Load program creates a new row in MRP_FORECAST_ITEMS for each new forecast entry that refers to an item that has not been assigned to the forecast referenced in the Open Forecast Interface.

Optional Data

Use WORKDAY_CONTROL to indicate the action that the Forecast Interface Load should take if it finds a forecast date or forecast end date that is not a valid workday. Enter one of the following:

- 1. Reject
- 2. Shift forward
- 3. Shift backward

If WORKDAY_CONTROL is set to Null, the Forecast Interface Load program assumes a value of 1 (Reject).

Use BUCKET_TYPE to indicate the bucket type of each new forecast entry. Enter one of the following:

- 1. Days
- 2. Weeks
- 3. Periods

If BUCKET_TYPE is null, the Forecast Interface Load program assumes a value of 1 (Days).

Use FORECAST_END_DATE for forecast entries that span multiple periods.

Use TRANSACTION_ID if you wish to replace an existing entry in MRP_FORECAST_DATES with a new forecast entry that you have loaded into the

Open Forecast Interface. The Forecast Interface Load deletes any existing entries in MRP_FORECAST_DATES with the same TRANSACTION_ID before importing the new forecast entry.

Use SOURCE_CODE and SOURCE_LINE_ID to identify the source of new forecast entries.

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

Validation

Standard Validation

Oracle Master Scheduling/MRP validates all required columns in the interface table. For specific information on the data implied by these columns, see your *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual* for details.

Other Validation

Oracle Master Scheduling/MRP also performs the following validation:

INVENTORY_ITEM_ID

Must be a valid item defined IN MTL_SYSTEM_ITEMS.

ORGANIZATION_ID

Must be a valid organization defined in ORG_ORGANIZATION_DEFINITIONS.

FORECAST_DESIGNATOR

Must be a valid, non-disabled forecast name defined in MRP_FORECAST_DESIGNATORS.

FORECAST_DATE

Must be less than or equal to RATE_END_DATE if RATE_END_DATE is provided.

FORECAST_END_DATE

Must be greater than or equal to FORECAST_DATE.

FORECAST_QUANTITY

Must be greater than 0 and less than or equal to 99999999.9.

PROCESS_STATUS

Must be one of:

- 1. Do not process
- 2. Waiting to be processed
- 3. Being processed
- 4. Error
- 5. Processed

WORKDAY_CONTROL

Must be one of:

- 1. Reject
- 2. Shift forward
- 3. Shift backward

BUCKET_TYPE

Must be one of:

- 1. Days
- 2. Weeks
- 3. Periods

TRANSACTION_ID

If provided, TRANSACTION_ID must match an existing TRANSACTION_ID in MRP_FORECAST_DATES.

Resolving Failed Open Forecast Interface Rows

Error Messages

Oracle Master Scheduling/MRP may display specific error messages during interface processing.

See Also

The *Oracle Applications Message Reference Manua*l. This manual is available in HTML format on the documentation CD-ROM for Release 11i.

Viewing Failed Transactions

Use SQL*Plus to view failed transactions in the Open Forecast Interface. The ERROR_MESSAGE column indicates why the Forecast Interface Load program was unable to successfully process each failed transaction.

Fixing Failed Transactions Options

Use SQL*Plus to manually correct failed transactions. You can either:

- Delete the failed row in the Open Forecast Interface, correct the error in your external forecast, and reload the corrected forecast into the Open Forecast Interface, or
- Correct the error in the Open Forecast Interface, reset the PROCESS_STATUS column to2 (Waiting to be processed), and set the REQUEST_ID and ERROR_MESSAGE columns to Null

The Planning Manager will detect the new rows when it next checks the Open Forecast Interface, and launch the Forecast Interface Load program accordingly.

Open Master Schedule Interface

You can import master schedules from any source using the Open Master Schedule Interface. Oracle Master Scheduling/MRP automatically validates and implements imported master schedules as new master schedules in Oracle Master Scheduling/MRP.

The purpose of this essay is to explain how to use the Open Master Schedule Interface so that you can integrate other applications with Oracle Master Scheduling/MRP.

Functional Overview

All processing is performed by the Master Schedule Interface Load program. The Master Schedule Interface Load program is launched by the Planning Manager, which periodically checks the Open Master Schedule Interface to see if there are any new rows waiting to be processed.

Setting Up the Open Master Schedule Interface

You must define at least one organization, item, and master schedule name before using the Open Master Schedule Interface. Since the Planning Manager decides when to call the Master Schedule Interface Load program, the Planning Manager must also be running before you can import master schedules via the Open Master Schedule Interface.

Inserting into the Open Master Schedule Interface Table

You must load your master schedules into the MRP_SCHEDULE_INTERFACE table. The Master Schedule Interface Load program validates your master schedules, derives

any additional data as necessary, and then processes it by creating new master schedules in Oracle Master Scheduling/MRP.

Open Master Schedule Interface Table Description

The Open Master Schedule Interface Table is described in the following table:

Column Name	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	Number	x		
SCHEDULE_DESIGNATOR	Varchar2(10)	x		
ORGANIZATION_ID	Number	x		
LAST_UPDATE_DATE	Date			x
LAST_UPDATED_BY	Number			x
CREATION_DATE	Date			x
CREATED_BY	Number			x
LAST_UPDATE_LOGIN	Number			x
SCHEDULE_DATE	Date	x		
NEW_SCHEDULE_DATE	Date		x	
RATE_END_DATE	Date			x
NEW_RATE_END_DATE	Date		x	

Column Name	Type	Required	Derived	Optional
SCHEDULE_QUANTITY	Number	x		
SCHEDULE_COMMENTS	Varchar2(240)			x
ERROR_MESSAGE	Varchar2(240)		x	
WORKDAY_CONTROL	Number			x
TRANSACTION_ID	Number			x
PROCESS_STATUS	Number	x		
SOURCE_CODE	Varchar2(10)			x
SOURCE_LINE_ID	Number			x
REQUEST_ID	Number		x	
PROGRAM_APPLICATION_ID	Number		x	
PROGRAM_ID	Number		x	
PROGRAM_UPDATE_DATE	Date		x	
ATTRIBUTE_CATEGORY	Varchar2(30)			x
ATTRIBUTE1 - ATTRIBUTE15	Varchar2(150)			x
PROJECT ID	Number(15)			x

Column Name	Type	Required	Derived	Optional
TASK ID	Number(15			x
LINE ID	Number(15			x

Legend

1: Rate-based Master Schedule Entries only

Required Data

ORGANIZATION_ID, SCHEDULE_DESIGNATOR, INVENTORY_ITEM_ID, SCHEDULE_DATE, and SCHEDULE_QUANTITY are used by the Master Schedule Interface Load program to create new schedule entries in MRP_SCHEDULE_DATES.

PROCESS_STATUS indicates the current state of processing of each new schedule entry in the Open Master Schedule Interface. Possible values include:

- 1. Do not process
- 2. Waiting to be processed
- 3. Being processed
- 4. Error
- 5. Processed

When you first load a new schedule entry into the Open Master Schedule Interface, set PROCESS_STATUS to 2 (Waiting to be processed).

Derived Data

ERROR_MESSAGE indicates a problem that prevents the Master Schedule Interface Load program from successfully processing a new schedule entry in the Open Master Schedule Interface.

The Master Schedule Interface Load program creates a new row in MRP_SCHEDULE_ITEMS for each new schedule entry that refers to an item that has not been assigned to the master schedule referenced in the Open Master Schedule Interface.

Optional Data

Use WORKDAY_CONTROL to indicate the action that the Master Schedule Interface Load should take if it finds a schedule date or schedule end date that is not a valid workday. Enter one of the following:

- 1. Reject
- 2. Shift forward
- 3. Shift backward

If WORKDAY_CONTROL is set to Null, the Master Schedule Interface Load program assumes a value of 1 (Reject).

Use SCHEDULE_END_DATE for rate-based master schedule entries.

Use TRANSACTION_ID if you wish to replace an existing entry in MRP_SCHEDULE_DATES with a new schedule entry that you have loaded into the Open Master Schedule Interface. The Master Schedule Interface Load deletes any existing entries in MRP_SCHEDULE_DATES with the same TRANSACTION_ID before importing the new schedule entry.

Use SOURCE_CODE and SOURCE_LINE_ID to identify the source of new schedule entries.

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

Validation

Standard Validation

Oracle Master Scheduling/MRP validates all required columns in the interface table. For specific information on the data implied by these columns, see your *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual* for details.

Other Validation

Oracle Master Scheduling/MRP also performs the following validation:

INVENTORY_ITEM_ID

Must be a valid item defined IN MTL_SYSTEM_ITEMS. Master Demand Schedules can only include MPS planned or MRP planned items. Master Production Schedules can only include MPS planned items.

ORGANIZATION_ID

Must be a valid organization defined in ORG_ORGANIZATION_DEFINITIONS.

SCHEDULE_DESIGNATOR

Must be a valid, non-disabled master schedule name defined in

MRP_SCHEDULE_DESIGNATORS.

SCHEDULE_DATE

Must be less than or equal to RATE_END_DATE if RATE_END_DATE is provided.

RATE_END_DATE

Must be greater than or equal to SCHEDULE_DATE.

Only repetitively planned items can have a RATE_END_DATE.

SCHEDULE_QUANTITY

Must be greater than 0 and less than or equal to 99999999.9.

WORKDAY_CONTROL

Must be one of:

- 1. Reject
- 2. Shift forward
- 3. Shift backward

PROCESS_STATUS

Must be one of:

- 1. Do not process
- 2. Waiting to be processed
- 3. Being processed
- 4. Error
- 5. Processed

TRANSACTION_ID

If provided, TRANSACTION_ID must match an existing TRANSACTION_ID in MRP_SCHEDULE_DATES.

Resolving Failed Open Master Schedule Interface Rows

Error Messages

Oracle Master Scheduling/MRP may display specific error messages during interface

processing.

See Also

*The Oracle Applications Message Reference Manua*l. This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Viewing Failed Transactions

Use Scalpels to view failed transactions in the Open Master Schedule Interface. The ERROR_MESSAGE column indicates why the Master Schedule Interface Load program was unable to successfully process each failed transaction.

Fixing Failed Transactions Options

Use Scalpels to manually correct failed transactions. You can either:

- Delete the failed row in the Open Master Schedule Interface, correct the error in your external schedule, and reload the corrected schedule into the Open Master Schedule Interface, or
- Correct the error in the Open Master Schedule Interface, reset the PROCESS_STATUS column to2 (Waiting to be processed), and set the REQUEST_ID and ERROR_MESSAGE columns to Null

The Planning Manager will detect the new rows when it next checks the Open Master Schedule Interface, and launch the Master Schedule Interface Load program accordingly.

Open Forecast Entries Application Program Interface

The Open Forecast Entries Application Program Interface(API) allows you to create, replace, or delete forecast entries for existing forecasts and forecast sets in Oracle Master Scheduling/MRP.

The purpose of this essay is to explain how to use the Open Forecast Entries API so that you can integrate other applications with Oracle Master Scheduling/MRP. The Open Forecast Entries API differs from the Open Forecast Interface in two ways:

- There is tighter coupling between the calling interface and the MRP system.
- It is used for synchronous actions on the forecasting data, and you can manipulate data within a commit cycle controlled by the calling module.

This is achieved by the use of a PL/SQL table instead of a database table.

Functional Overview

You can process MRP Forecast Entries directly from within your calling module without running a concurrent process, it is PL/SQL based. This program allows you to

create new forecasts, replace existing forecasts, and delete forecast entries within a defined forecast name or designator. The forecast data that needs to be imported is loaded from a table and inserted into the MRP_FORECAST_DATES parameter.

Setting Up the Open Forecast Entries API

The Open Forecast Entries API is a stored PL/SQL function, **MRP_FORECAST_INTERFACE_PK.MRP_FORECAST_INTERFACE,**with two parameters. One parameter is a PL/SQL table structured the same as MRP_FORECAST_INTERFACE. The second parameter is a table defining the forecast and organization.

Inserting into the Open Forecast Entries API Tables

You must load your forecast data into the T_FORECAST_INTERFACE PL/SQL table, and the FORECAST_DESIGNATOR PL/SQL table.

Open Forecast Entries Application Program Interface PL/SQL Table Description

The Open Forecast Entries Application Program Interface PL/SQL table is described in the following table:

T_FORECAST_INTERFACEColumn Name	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	Number	x		
FORECAST_DESIGNATOR	Varchar2(10)	x		
ORGANIZATION_ID	Number	x		
FORECAST_DATE	Date	x		
LAST_UPDATE_DATE	Number		x	
CREATION_DATE	Date		x	
CREATED_BY	Number		x	

T_FORECAST_INTERFACEColumn Name	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	Number		x	x
QUANTITY	Number	x		
PROCESS_STATUS	Number	x		
CONFIDENCE_PRECENTAGE	Number	x		
COMMENTS	Varchar2(240)			x
ERROR_MESSAGE	Varchar2(240)		x	
REQUEST_ID	Number		x	
PROGRAM_APPLICATION_ID	Number		x	
PROGRAM_ID	Number		x	
PROGRAM_UPDATE_DATE	Date		x	
WORKDAY_CONTROL	Number			x
BUCKET_TYPE	Number			x
FORECAST_END_DATE	Date			x
TRANSACTION_ID	Number			x
SOURCE_CODE	Varchar2(10)			x

T_FORECAST_INTERFACEColumn Name	Type	Required	Derived	Optional
SOURCE_LINE_ID	Number			x
ATTRIBUTE1 - ATTRIBUTE15	Varchar2(150)			x
PROJECT ID	Number(15)			x
TASK ID	Number(15			x
LINE ID	Number(15			x

Open Forecast Interface Designator Table Description

The Open Forecast Interface Designator Table is described in the following table:

T_FORECAST_DESIGNATOR Column Name	Type	Required	Derived	Optional
ORGANIZATION_ID	Number	x		
FORECAST_DESIGNATOR	Varchar2(10)	x		

Legend

1: Multiple Period Forecast Entries only

Returns

True if successful.

False if failure.

Parameters

Name	Type	In/Out
T_FORECAST_INTERFACE	Table of MRP_FORECAST_INTERFACE ROWTYPE	In and Out
T_FORECAST_DESIGNATOR	Table of User-defined record REC_FORECAST_DESG (Organization_id number, Forecast Designator Varchar2(10)	In

Required Data

ORGANIZATION_ID, FORECAST_DESIGNATOR, INVENTORY_ITEM_ID, FORECAST_DATE, and QUANTITY are used by the Forecast Interface Entries program to create, replace, or delete forecast entries in T_FORECAST_INTERFACE.

PROCESS_STATUS indicates the current state of processing of each new forecast entry. Valid values include:

- 1. Do not process
- 2. Waiting to be processed

When you first load a new forecast entry into the Open Forecast Entries Interface, set PROCESS_STATUS to 2 (Waiting to be processed). The values 3 (Being processed), 4 (Error), and 5 (Processed) are used to report back to the calling program.

Derived Data

The concurrent program and WHO columns, along with the error message column, are derived and set by the API accordingly.

Optional Data

Use WORKDAY_CONTROL to indicate the action that the Forecast Interface Entry should take if it finds a forecast date or forecast end date that is not a valid workday. Enter one of the following:

- 1. Reject
- 2. Shift forward
- 3. Shift backward

If WORKDAY_CONTROL is set to Null, the Forecast Interface Entry program assumes a value of 1 (Reject).

Use BUCKET_TYPE to indicate the bucket type of each new forecast entry. Enter one of the following:

- 1. Days
- 2. Weeks
- 3. Periods

If BUCKET_TYPE is null, the Forecast Interface Load program assumes a value of 1 (Days).

Use FORECAST_END_DATE for forecast entries that span multiple periods.

Use TRANSACTION_ID if you wish to replace an existing entry in MRP_FORECAST_DATES with a new forecast entry that you have loaded into the Open Forecast Interface. The Forecast Interface Load deletes any existing entries in MRP_FORECAST_DATES with the same TRANSACTION_ID before importing the new forecast entry.

Use SOURCE_CODE and SOURCE_LINE_ID to identify the source of new forecast entries.

See Also

Oracle Master Scheduling/MRP Technical Reference Manual

Validation

Standard Validation

Oracle Master Scheduling/MRP validates all required columns in the interface table. For specific information on the data implied by these columns, see your *Oracle Master Scheduling/MRP Reference Manual* for details.

Other Validation

Oracle Open Forecast Entries Interface also performs the following validation:

INVENTORY_ITEM_ID

Must be a valid item defined IN MTL_SYSTEM_ITEMS.

FORECAST_DESIGNATOR

Must be a valid, non-disabled forecast name defined in MRP_FORECAST_DESIGNATORS.

ORGANIZATION_ID

Must be a valid organization defined in ORG_ORGANIZATION_DEFINITIONS.

FORECAST_DATE

Must be less than or equal to FORECAST_END_DATE if FORECAST_END_DATE is provided.

PROCESS_STATUS

Must be one of:

- 1. Do not process
- 2. Waiting to be processed

FORECAST_END_DATE

Must be greater than or equal to FORECAST_DATE.

Must be greater than 0 and less than or equal to 99999999.9.

FORECAST_QUANTITY

Must be greater than 0 and less than or equal to 99999999.9.

WORKDAY_CONTROL

Must be one of:

- 1. Reject
- 2. Shift forward
- 3. Shift backward

BUCKET_TYPE

Must be one of:

- 1. Days
- 2. Weeks
- 3. Periods

TRANSACTION_ID

If provided, TRANSACTION_ID must match an existing TRANSACTION_ID in MRP_FORECAST_DATES.

Using the Open Forecast Entries API

Creating New Forecast Entries

Populate table T_FORECAST_INTERFACE with all the forecast data that needs to be imported. Set PROCESS_STATUS to a value of 2 for all rows.Call MRP_FORECAST_INTERFACE_PK.MRP _FORECAST_INTERFACE using parameter T_FORECAST_INTERFACE and T_FORECAST_DESIGNATOR. The Forecast Interface Entry program creates a new row in MRP_FORECAST_ITEMS for each new forecast entry that refers to an item that has not been assigned to the forecast referenced in the Open Forecast Interface. The application program interface will process the rows and set the column PROCESS_STATUS to a value of either 4 or 5:

- 4 an error occurred, the column ERROR_MESSAGE will indicate the error
- 5 the row was inserted into MRP_FORECAST_DATES

Replacing Forecast Entries

- Populate table T_FORECAST_INTERFACE with all the forecast data that needs to be imported. Set PROCESS_STATUS to a value of 2 for all rows.
- Populate table T_FORECAST_DESIGNATOR with all the forecast desIgnators for which entries need to be deleted.
- Call MRP_FORECAST_INTERFACE_PK.MRP_FORECAST_INTERFACE with the following parameters: FORECAST_INTERFACE, T_FORECAST_DESIGNATOR.
- The application program interface will delete the existing entries for each forecast designator in T_FORECAST_DESIGNATOR. It will process the rows in T_FORECAST_INTERFACE and set the column PROCESS_STATUS to a value of either 4 or 5:
  - 4 an error occurred, the column ERROR_MESSAGEwill indicate the error
  - 5 the row was inserted into MRP_FORECAST_DATES

Deleting All Forecast Entries in Multiple Forecast Designators

- Populate table T_FORECAST_DESIGNATOR with all the forecast desIgnators for which entries need to be deleted.
- Call MRP_FORECAST_INTERFACE_PK.MRP _FORECAST_INTERFACE with parameter T_FORECAST_DESIGNATOR.
- The application program interface will delete the existing entries for each forecast

desIgnator in T_FORECAST_DESIGNATOR.

Error Handling

The Open Forecast Entries Interface program will process the rows and report the following values for every record in the FORECAST__INTERFACE entry table.

Condition	PROCESS_STATUS	ERROR_MESSAGE
SuccessFailure	54	Nullactual error message

Sourcing Rule Application Program Interface

The Sourcing Rule Application Program Interface (API) is a public API that allows you to create, maintain, and delete sourcing rule or bill of distribution information in Oracle Master Scheduling/MRP and Oracle Supply Chain Planning.

This API differs from the planning interfaces because it puts information directly into the Oracle Master Scheduling/MRP tables rather than inserting information into an interface table. You can process sourcing rule entries directly from within your calling module without running a concurrent process. It is PL/SQL based, you can process one sourcing rule or bill of distribution per call.

It can be used for both custom applications and legacy systems. The Sourcing Rule API consists of two objects: the Sourcing Rule object and the Assignment object. Each of these objects consists of several entities.

This section explains how to use the Sourcing Rule API and how it functions in Oracle planning products.

Sourcing Rule/Bill of Distribution API Features

The Sourcing Rule/Bill of Distribution API object consists of three entities: (mfgapi_7_1.gif)

- Sourcing Rule/Bill of Distribution

  You can create new entries, update existing sourcing rule/bill of distribution information, and delete entries.

  Table: MRP_SOURCING_RULES

- Receiving Organization

  You can process multiple receiving organizations belonging to the sourcing rule/bill of distribution. You can create new entries, update existing information, and delete receiving organizations.

Table: MRP_SR_RECEIPT_ORG

- Shipping Organization

  You can process multiple shipping organizations belonging to the sourcing rule/bill of distribution. You can create new entries, update existing information, and delete shipping organizations.

  Table: MRP_SR_SOURCE_ORG

The relationships between these tables create the sourcing information used in MPS, MRP, and DRP plans. The MRP_SOURCING_RULES table stores sourcing rule names and descriptions. It contains receiving organization data for material sources. The receiving organization information is in the MRP_SR_RECEIPT_ORG table, each row of the table specifies a receiving organization for a date range. The MRP_SR_SOURCE_ORG table stores data on the source suppliers for the sourcing rule or bill of distribution.

A sourcing rule is paired with an assignment from the MRP_SR_ASSIGNMENTS table, all this information is fed into the material items and categories tables.

For more information on planning table, see: *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual*.

Functional Overview

The Sourcing Rule/Bill of Distribution API provides three public procedures for calling the create, update, delete, and get operations:

- Process_Sourcing_Rule

  Accepts object specific information (through the parameters) and handles Create, Update and Delete operation.

- Get_Sourcing_Rule

  Handles the Select Operation Lock_Sourcing_Rule.

- Select Operation Lock_Sourcing_Rule

  Locks records that define a particular Sourcing Rule and associated child entities.

Each of these three procedures first performs a check for call compatibility and then calls the respective private API. There is a specific order of processing information into the parameters and it is as follows:

- First, Sourcing Rule information is processed by passing in through the p_sourcing _rule_rec parameter.

- Next, the receiving organization information is passed to the p_receiving_org parameter.

- And then the shipping organization information is processed through the p_shipping_org parameter.

Setting Up the Sourcing Rule/Bill of Distribution API

The Sourcing Rule API is a stored PL/SQL function. Before using the API, set up and activate the following parameters:

- Version number
- Sourcing rule
- Receiving organization
- Shipping organization

Procedure Parameter Descriptions

MRP_SOURCING_RULE_PUB.PROCESS_SOURCING_RULE

The following chart describes all parameters used by the public API MRP_SOURCING_RULE_PUB.PROCESS_SOURCING_RULE procedure. All of the inbound and outbound parameters are listed. Additional information on these parameters follows.

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_return_values	IN	Varchar2			x
p_commit	IN	Varchar2			x
x_return_status	OUT	Varchar2			x
x_msg_count	OUT	Number			x
x_msg_data	OUT	Varchar2		x	

Parameter	Usage	Type	Required	Derived	Optional
p_sourcing_rule_rec	IN	Record	x		
p_sourcing_rule_val_rec	IN	Record	x		
p_receiving_org_tbl	IN	PL/SQL Table	x		
p_receiving_org_val_tbl	IN	PL/SQL Table	x		
p_shipping_org_tbl	IN	PL/SQL Table	x		
p_shipping_org_val_tbl	IN	PL/SQL Table	x		
x_sourcing_rule_rec	OUT	Record			x
x_sourcing_rule_val_rec	OUT	Record			x
x_receiving_org_tbl	OUT	PL/SQL Table			x
x_receiving_org_val_tbl	OUT	PL/SQL Table			x
x_shipping_org_tbl	OUT	PL/SQL Table			x
x_shipping_org_val_tbl	OUT	PL/SQL Table		x	

p_api_version_number

Used to compare the incoming API call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API send back the values on your behalf.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates number of error messages API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_sourcing_rule_rec

The sourcing rule or bill of distribution record referenced by the API.

Default Value: G_MISS_SOURCING_RULE_REC

p_sourcing_rule_val_rec

Resolves the values for the API, and then returns the information for the sourcing rule/bill of distribution record.

Default Value: G_MISS_SOURCING_RULE_VAL_REC

p_receiving_org_tbl

The receiving organization information listed in the rule is returned to this parameter.

Default Value: G_MISS_RECEIVING_ORG_TBL

p_receiving_org_val_tbl

Resolves the values for the API, and then returns the information for the receiving organization listed in the sourcing rule.

Default Value: G_MISS_RECEIVING_ORG_VAL_TBL

p_shipping_org_tbl

The shipping organization information listed in the rule is returned to this parameter.

Default Value: G_MISS_SHIPPING_ORG_TBL

p_shipping_org_val_tbl

Resolves the values for the API, and then returns the information for the shipping organization listed in the sourcing rule.

Default Value: G_MISS_SHIPPING_ORG_VAL_TBL

x_sourcing_rule_rec

Result of the API data after it completes its function for the sourcing rule/bill of distribution record.

x_sourcing_rule_val_rec

Resolves the values for the API, and then returns the information for the sourcing rule/bill of distribution record.

x_receiving_org_tbl

Resolves the values for the API, and then returns the information for the receiving organization listed in the sourcing rule/bill of distribution record.

x_receiving_org_val_tbl

Resolves the values for the API, and then returns the information for the receiving organization listed in the rule.

x_shipping_org_tbl

Resolves the values for the API, and then returns the information for the shipping organization listed in the sourcing rule/bill of distribution record.

x_shipping_org_val_tbl

Resolves the values for the API, and then returns the information for the shipping organization listed in the rule.

Record Parameter Descriptions

SOURCING_RULE_REC_TYPE

The procedure passes information to record groups and PL/SQL tables. The chart below describes all records that are used by the SOURCING_RULE_REC_TYPE record. Additional information on these parameters follows.

Parameter	Type	Required	Derived	Optional
sourcing_rule_id	Number	x update, delete		
attribute 1 - 15	Varchar2(150)			x
attribute_category	Varchar2(30)			x
created_by	Number		x	
creation_date	Date		x	
description	Varchar2(80)			x
last_updated_by	Number		x	
last_update_date	Date		x	
last_update_login	Number		x	

Parameter	Type	Required	Derived	Optional
organization_id	Number	x		
planning_active	Number			x
program_application_id	Number			x
program_id	Number		x	
program_update_date	Date		x	
request_id	Number			x
sourcing_rule_name	Varchar2(30)	x insert		x
sourcing_rule_type	Number	x		x
status	Number			x
return_status	Varchar2(1)			x
db_flag	Varchar2(1)		x	
operation	Varchar2(30)		x	

sourcing_rule_id

Identification number for the sourcing rule or bill of distribution record referenced by the API.

Default Value: FND_API.G_MISS_NUM

attribute 1 - 15

Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

attribute_category

The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

created_by

Identification number for user initiating this program session.

Default Value: FND_API.G_MISS_NUM

creation_date

Date this program session was created.

Default Value: FND_API.G_MISS_DATE

description

Text describing the sourcing rule record type.

Default Value: FND_API.G_MISS_CHAR

last_updated_by

User ID for user creating this program session.

Default Value: FND_API.G_MISS_NUM

last_update_date

Date program was last updated.

Default Value: FND_API.G_MISS_DATE

last_update_login

User login for user updating this program.

Default Value: FND_API.G_MISS_NUM

organization_id

The identification number for the organization referenced in the sourcing rule or bill of distribution record.

Default Value: FND_API.G_MISS_NUM

planning_active

Rule is active when the sum of the allocation percentages equals 100.

Default Value: FND_API.G_MISS_NUM

program_application_id

Application identifier of the program that has made a call to the Sourcing Rule API if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_id

Identifier of the program that has made a call to the Sourcing Rule API, if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_update_date

The date when the program inserts or updates the sourcing records into the appropriate tables.

Default Value: FND_API.G_MISS_DATE

request_id

The request ID determines which profile values are used as a default.

Default Value: FND_API.G_MISS_NUM

sourcing_rule_name

Valid name of rule defined in the MRP_SOURCING _RULES table.

Default Value: FND_API.G_MISS_CHAR

sourcing_rule_type

Valid types must be one of the following values:

- (1) Sourcing Rule
    - (2) Bill of Distribution

Default Value: FND_API.G_MISS_NUM

status

If any validation fails, the API will return error status to the calling module. The Sourcing Rule API processes the rows and reports the following values for every record. If the sourcing rule does not already exist in the system - the Status attribute is set to 1.

Processing status of the sourcing rule, valid values are:

- (1)
- (2)

Default Value: FND_API.G_MISS_NUM

return_status

Processing status of the API after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Default Value: FND_API.G_MISS_CHAR

db_flag

Indicator of the record existing in the database.

Default Value: FND_API.G_MISS_CHAR

operation

Indicator of whether the record is inserted, updated, or deleted. Valid values include:

- Create
- Update
- Delete

Default Value: FND_API.G_MISS_CHAR

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

RECEIVING_ORG_REC_TYPE

The procedure passes information to record groups and PL/SQL tables. The chart below describes all records that are used by the RECEIVING_ORG_REC_TYPE record. Additional information on these parameters follows.

Parameter	Type	Required	Derived	Optional
sr_receipt_id	Number	x		
attribute 1 - 15	Varchar2(150)			x

Parameter	Type	Required	Derived	Optional
attribute_category	Varchar2(30)			x
created_by	Number		x	
creation_date	Date		x	
disable_date	Date		x	
effective_date	Date		x	
last_updated_by	Number		x	
last_update_date	Date		x	
last_update_login	Number		x	
program_application_id	Number		x	
program_id	Number		x	
program_update_date	Date		x	
receipt_organization_id	Number	x		
request_id	Number		x	
sourcing_rule_id	Number	x		
return_status	Varchar2(1)			x
db_flag	Varchar2(1)		x	
operation	Varchar2(30)		x	

sr_receipt_id

Identification number for the receiving organization referenced in the sourcing rule or bill of distribution record.

Default Value: FND_API.G_MISS_NUM

attribute 1 - 15

Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

attribute_category

The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

created_by

Identification number for user initiating this program session.

Default Value: FND_API.G_MISS_NUM

creation_date

Date this program session was created.

Default Value: FND_API.G_MISS_DATE

disable_date

Date the receipt organization is no longer effective.

Default Value: FND_API.G_MISS_DATE

effective_date

Beginning date the receipt organization becomes effective.

Default Value: FND_API.G_MISS_DATE

last_updated_by

User ID for user creating this program session.

Default Value: FND_API.G_MISS_NUM

last_update_date

Date program was last updated.

Default Value: FND_API.G_MISS_DATE

last_update_login

User login for user updating this program.

Default Value: FND_API.G_MISS_NUM

program_application_id

Application identifier of the program that has made a call to the Sourcing Rule API if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_id

Identifier of the program that has made a call to the Sourcing Rule API, if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_update_date

The date when the program inserts or updates the sourcing records into the appropriate tables.

Default Value: FND_API.G_MISS_DATE

receipt_organization_id

Identifier of the organization that serves as the destination for the sourcing rule or bill of distribution.

Default Value: FND_API.G_MISS_NUM

request_id

The request ID determines which profile values are used as a default.

Default Value: FND_API.G_MISS_NUM

sourcing_rule_id

Identification number for the sourcing rule or bill of distribution record referenced by the API.

Default Value: FND_API.G_MISS_NUM

return_status

Processing status of the API after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR

- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Default Value: FND_API.G_MISS_CHAR

db_flag

Indicator of the record existing in the database.

Default Value: FND_API.G_MISS_CHAR

operation

Indicator of whether the record is inserted, updated, or deleted. Valid values include:

- Create
- Update
- Delete

Default Value: FND_API.G_MISS_CHAR

SHIPPING_ORG_REC_TYPE

The procedure passes information to record groups and PL/SQL tables. The chart below describes all records that are used by the SHIPPING_ORG_REC_TYPE record. Additional information on these parameters follows.

Parameter	Type	Required	Derived	Optional
sr_source_id	Number	x		
allocation_percent	Number	x		
attribute 1 - 15	Varchar2(150)			x
attribute_category	Varchar2(30)			x
created_by	Number		x	
creation_date	Date		x	
last_updated_by	Number		x	
last_update_date	Date		x	

Parameter	Type	Required	Derived	Optional
last_update_login	Number		x	
program_application_id	Number		x	
program_id	Number		x	
program_update_date	Date		x	
rank	Number	x		
request_id	Number		x	
secondary_inventory	Varchar2(10)		x	
ship_method	Varchar2(30)	x		
source_organization_id	Number	x		
source_type	Number			
sr_receipt_id	Number			
vendor_id	Number			
vendor_site_id	Number			
return_status	Varchar2(1)			x
db_flag	Varchar2(1)		x	
operation	Varchar2(30)		x	
receiving_org_index	Number			

sr_source_id

Primary key in the sourcing rule or bill of distribution table.

Default Value: FND_API.G_MISS_NUM

allocation_percent

Percentage allocated to each source organization/supplier site destination.

Default Value: FND_API.G_MISS_NUM

attribute 1 - 15

Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

attribute_category

The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

created_by

Identification number for user initiating this program session.

Default Value: FND_API.G_MISS_NUM

creation_date

Date this program session was created.

Default Value: FND_API.G_MISS_DATE

last_updated_by

User ID for user creating this program session.

Default Value: FND_API.G_MISS_NUM

last_update_date

Date program was last updated.

Default Value: FND_API.G_MISS_DATE

last_update_login

User login for user updating this program.

Default Value: FND_API.G_MISS_NUM

program_application_id

Application identifier of the program that has made a call to the Sourcing Rule API if it

is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_id

Identifier of the program that has made a call to the Sourcing Rule API, if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_update_date

The date when the program inserts or updates the sourcing records into the appropriate tables.

Default Value: FND_API.G_MISS_DATE

rank

Rank of the sources, valid values are non-zero integers.

Default Value: FND_API.G_MISS_NUM

request_id

The request ID determines which profile values are used as a default.

Default Value: FND_API.G_MISS_NUM

secondary_inventory

Not currently used.

ship_method

Method used when transporting material between source and destination.

Default Value: FND_API.G_MISS_CHAR

source_organization_id

Identifier of the source organization.

Default Value: FND_API.G_MISS_NUM

source_type

Indicator of the type of source. Valid values are:

- Make
- Transfer
- Buy

Default Value: FND_API.G_MISS_NUM

sr_receipt_id

Identification number for the receiving organization referenced in the sourcing rule or bill of distribution record.

Default Value: FND_API.G_MISS_NUM

vendor_id

Identifier of the vendor suppling the materials.

Default Value: FND_API.G_MISS_NUM

vendor_site_id

Identifier where the vendor's materials are located.

Default Value: FND_API.G_MISS_NUM

return_status

Processing status of the API after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Default Value: FND_API.G_MISS_CHAR

db_flag

Indicator of the record existing in the database.

Default Value: FND_API.G_MISS_CHAR

operation

Indicator of whether the record is inserted, updated, or deleted. Valid values include:

- Create
- Update
- Delete

Default Value: FND_API.G_MISS_CHAR

receiving_org_index

Foreign key to the receipt organization PL/SQL table.

Default Value: FND_API.G_MISS_NUM

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

Validation of Sourcing Rule /Bill of Distribution API

Standard Validation

Oracle Master Scheduling/MRP validates all required columns in the Sourcing Rule/Bill of Distribution API. For specific information on the data implied by these columns, see your Oracle Master Scheduling/MRP Technical Reference Manual for details.

If you do not want to update a particular column in:

- MRP_SOURCING_RULE_PUB.PROCESS_SOURCING_RULE

do not enter NULL for its corresponding interface parameter unless the default in the PL/SQL specification is NULL. Either use one of the missing parameter constants defined in the FND_API package (G_MISS_...), or do not pass any value at all.

For all flag parameters, pass in a Boolean constant defined in FND_API (G_TRUE or G_FALSE).

Each time the API is called, it will check the allocation percent for each receiving organization that belongs to the sourcing rule or bill of distribution. If the total allocation percent is 100, the plannning_active attribute is set to a value of 1. Otherwise the attribute is set to 2.

Creating Sourcing Rule API Entries

When you create a new sourcing rule the following item level validations:

- sourcing_rule_name: must be defined in the MRP_SOURCING _RULES table.
- sourcing_rule_type: must be a either (1) Sourcing Rule or (2) Bill of Distribution.If the sourcing rule does not already exist in the system - the Status attribute is set to 1.

When you create a new sourcing rule, the following record level validations occur:

- organization_id: must be a valid organization defined in ORG_ORGANIZATION_DEFINITIONS.
- The organization_id attribute is associated with a valid organization, unless it is null.

When you create a new sourcing rule, the following object level validations occur:

- At least one receiving organization record is created.
- At least one shipping organization record is created.

If validation is successful, a record is inserted into the MRP_SOURCING_RULES table.

Updating Sourcing Rule API Entries

When you update an existing sourcing rule, the following item level validations occur:

- If the sourcing rule name is changed, the new name cannot already exist in the system.
- The organization cannot be changed and the sourcing rule type cannot be changed.

When you update an existing sourcing rule, the following record level validations occur:

- Required attributes are either sourcing_rule_id, or sourcing _rule_name, and organization_id, and all flexfields must be validated.

If validation is successful, a record is updated in the MRP_SOURCING_RULES table.

Deleting Sourcing Rule API Entries

You cannot delete a sourcing rule if assignment data exists for the rule. When you delete an existing sourcing rule, the following record level validation occurs:

- Required attributes are either sourcing_rule_id, or sourcing _rule_nam,e and organization_id.

When you delete an existing sourcing rule, the following object level validations occur:

- All receiving organization records associated with the rule/bill of distribution record in the MRP_SR_RECEIPT_ORG table.
- All shipping organization records associated with the rule/bill of distribution record in the MRP_SR_SOURCE_ORG table.
- The sourcing rule/bill of distribution record from MRP_SOURCING_RULES table.

If deletion is successful, the API returns a success status to the calling module.

Error Handling

If any validation fails, the API will return error status to the calling module. The Sourcing Rule API processes the rows and reports the following values for every record.

Condition	PROCESS_STATIS	ERROR_MESSAGE
Success	5	null
Failure	4	actual error message

See Also

Oracle Applications Message Reference Manual

Sourcing Rule Assignment API Features

The Sourcing Rule Assignment API object consists of two entities. The following chart demonstrates the relationship between the assignment object and the sourcing rule /bill of distribution object: (mfgapi_7_2.gif)

- Assignment Set

  You can create new entries, update existing assignment information, and delete entries.

  Table: MRP_ASSIGNMENT_SETS

- Assignment

  You can process multiple assignments belonging to the sourcing rule/bill of distribution. This includes creating new, updating or deleting existing entries.

  Table: MRP_SR_ASSIGNMENTS

The relationship between these tables create the sourcing assignment information used in MPS, MRP, and DRP plans. Once you have defined your sourcing rules and bills of distribution, you must assign them to items and organizations. These assignments are grouped together in sets. The MRP_ASSIGNMENT_SET table stores assignment set names and the different levels of assignment. For example, you may assign an items in all organizations, or just in an inventory organization. The MRP_SR_ASSIGNMENTS table stores data on the sourcing rule or bill of distribution for the assignment.

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

Functional Overview

The Sourcing Assignment API provides three public procedures for calling the create, update, delete and get operations:

- Process_Assignment

  Accepts object specific information (through the parameters) and handles Create, Update and Delete Operation.

- Get_Assignment

  Handles the Select Operation Lock_Assignment.

- Select Operation Lock_Assignment

  Locks records that define a particular Sourcing Rule and associated child entities.

Each of these three procedures first performs a check for call compatibility and then calls the respective private API. There is a specific order of processing information into the parameters and it is as follows:

- Assignment set information is processed by passing in through the p_assignment_set_rec table parameter.
- Next, the assignment information is passed to the p_assignment_set_tbl table parameter.

Setting Up the Sourcing Rule Assignment API

The Sourcing Rule Assignment API is a stored PL/SQL function. You need to define certain data before you create or update assignment information. Before using the API, set up and/or activate the following parameters:

- Version number
- Sourcing Assignment Set Number
- Assignment records

Procedure Parameter Descriptions

MRP_SRC_ASSIGNMENT_PUB.PROCESS_ASSIGNMENT

The table below describes all parameters that are used by the public API MRP_SRC_ASSIGNMENT_PUB.PROCESS_ASSIGNMENTprocedure. All of the inbound and outbound parameters are listed. Additional information on these parameters follows.

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_return_values	IN	Varchar2			x
p_commit	IN	Varchar2			x
x_return_status	OUT	Varchar2			x
x_msg_count	OUT	Number			x
x_msg_data	OUT	Varchar2		x	
p_assignment_set_rec	IN	Record	x		
p_assignment_set_val_rec	IN	Record	x		
p_assignment_tbl	IN	PL/SQL Table	x		
p_assignment_val_tbl	IN	PL/SQL Table	x		
x_assignment_set_rec	OUT	Record			x
x_assignment_set_val_rec	OUT	Record			x
x_assignment_tbl	OUT	PL/SQL Table			x
x_assignment_val_tbl	OUT	PL/SQL Table			x

p_api_version_number

Used to compare the incoming API call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API send back the values on your behalf.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates number of error messages API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_assignment_set_rec

Enter the assignment set record.

Default Value: G_MISS_ASSIGNMENT_SET_REC

p_assignment_set_val_rec

Resolves the values for the API, and then returns the information for the assignment set

listed in the sourcing rule/bill of distribution record.

Default Value: G_MISS_ASSIGNMENT_SET_VAL_REC

p_assignment_tbl

References the assignment parameters listed in the assignment set.

Default Value: G_MISS_ASSIGNMENT_TBL

p_assignment_val_tbl

Resolves the values for the API, and then returns the information for the assignment set listed in the rule.

Default Value: G_MISS_ASSIGNMENT_VAL_TBL

x_assignment_set_rec

The assignment set record.

x_assignment_set_val_rec

Resolves the values for the API, and then returns the information for the assignment set listed in the sourcing rule/bill of distribution record.

x_assignment_tbl

References the assignment listed in the assignment set.

Resolves the values for the API, and then returns the information for the assignment set listed in the sourcing rule/bill of distribution record.

x_assignment_val_tbl

Resolves the values for the API, and then returns the information for the assignment set listed in the rule.

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

Record Parameter Descriptions

ASSIGNMENT_SET_REC_TYPE

The procedure passes information to record groups and PL/SQL tables. The table below describes all record parameters that are used by the ASSIGNMENT_SET_REC_TYPE record. Additional information on these parameters follows.

Parameter	Type	Required	Derived	Optional
assignment_set_id	Number	x		
assignment_set_name	Varchar2(30)	x		
attribute 1 - 15	Varchar2(150)			x
attribute_category	Varchar2(30)			x
created_by	Number		x	
creation_date	Date		x	
description	Varchar2(80)			x
last_updated_by	Number		x	
last_update_date	Date		x	
last_update_login	Number		x	
program_application_id	Number		x	
program_id	Number		x	
program_update_date	Date		x	
request_id	Number		x	
return_status	Varchar2(1)		x	
db_flag	Varchar2(1)		x	
operation	Varchar2(30)	x		

assignment_set_id

Identification number for the assignment set record referenced by the API.

Default Value: FND_API.G_MISS_NUM

assignment_set_name

Valid name of the assignment set defined in the MRP_ASSIGNMENT_SETS table.

Default Value: FND_API.G_MISS_CHAR

attribute 1 - 15

Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

attribute_category

The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

created_by

Identification number for user initiating this program session.

Default Value: FND_API.G_MISS_NUM

creation_date

Date this program session was created.

Default Value: FND_API.G_MISS_DATE

description

Text describing the sourcing rule record type.

Default Value: FND_API.G_MISS_CHAR

last_updated_by

User ID for user creating this program session.

Default Value: FND_API.G_MISS_NUM

last_update_date

Date program was last updated.

Default Value: FND_API.G_MISS_DATE

last_update_login

User login for user updating this program.

Default Value: FND_API.G_MISS_NUM

program_application_id

Application identifier of the program that has made a call to the Sourcing Rule API if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_id

Identifier of the program that has made a call to the Sourcing Rule API, if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_update_date

The date when the program inserts or updates the sourcing records into the appropriate tables.

Default Value: FND_API.G_MISS_DATE

request_id

The request ID determines which profile values are used as a default.

Default Value: FND_API.G_MISS_NUM

return_status

Processing status of the API after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Default Value: FND_API.G_MISS_CHAR

db_flag

Indicator of the record existing in the database.

Default Value: FND_API.G_MISS_CHAR

operation

Indicator of whether the record is inserted, updated, or deleted. Valid values include:

- Create
- Update

- Delete

Default Value: FND_API.G_MISS_CHAR

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference Manual

ASSIGNMENT_REC_TYPE

The procedure passes information to record groups and PL/SQL tables. The table below describes all record parameters that are used by the ASSIGNMENT_REC_TYPE record. Additional information on these parameters follows.

Parameter	Type	Required	Derived	Optional
assignment_id	Number	x		
assignment_set_id	Number	x		
assignment_type	Number	x		
attribute 1 - 15	Varchar2(150)	x		
attribute_category	Varchar2(30)			
category_id	Number			
category_set_id	Number			
created_by	Number			
creation_date	Date			
customer_id	Number			
inventory_item_id	Number			
last_updated_by	Number			
last_update_date	Date			

Parameter	Type	Required	Derived	Optional
last_update_login	Number			
organization _id	Number			
program_application_id	Number			
program_id	Number			
program_update_date	Date			
request_id	Number			
secondary_inventory	Varchar2(10)			
ship_to_site_id	Number			
sourcing_rule_id	Number			
sourcing_rule_type	Number			
return_status	Varchar2(1)			
db_flag	Varchar2(1)			
operation	Varchar2(30)	x		

assignment_id

Identification number for the assignment record referenced by the API.

Default Value: FND_API.G_MISS_NUM

assignment_set_id

Identification number for the assignment set record referenced by the API.

Default Value: FND_API.G_MISS_NUM

assignment_type

Valid types of sourcing assignments include the following values:

- (1) Global
  - (2) Category
  - (3) Item
  - (4) Organization
  - (5) Category/Organization
  - (6) Item/Organization

Default Value: FND_API.G_MISS_NUM

attribute 1 - 15

Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

attribute_category

The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

created_by

Identification number for user initiating this program session.

Default Value: FND_API.G_MISS_NUM

creation_date

Date this program session was created.

Default Value: FND_API.G_MISS_DATE

customer_id

Identification number for the customer record referenced by the API.

Default Value: FND_API.G_MISS_NUM

inventory_item_id

Identification number for the item record referenced by the API.

Default Value: FND_API.G_MISS_NUM

last_updated_by

User ID for user creating this program session.

Default Value: FND_API.G_MISS_NUM

last_update_date

Date program was last updated.

Default Value: FND_API.G_MISS_DATE

last_update_login

User login for user updating this program.

Default Value: FND_API.G_MISS_NUM

organization_id

The identification number for the organization referenced in the assignment record.

program_application_id

Application identifier of the program that has made a call to the Sourcing Rule API if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_id

Identifier of the program that has made a call to the Sourcing Rule API, if it is registered as a concurrent program in Oracle Application Object Library.

Default Value: FND_API.G_MISS_NUM

program_update_date

The date when the program inserts or updates the sourcing records into the appropriate tables.

Default Value: FND_API.G_MISS_DATE

request_id

The request ID determines which profile values are used as a default.

Default Value: FND_API.G_MISS_NUM

secondary_inventory

Currently not used.

ship_to_site_id

Used with customer identification attribute and organization assignment type to define

shipping location.

Default Value: FND_API.G_MISS_NUM

sourcing_rule_id

Identification number for the sourcing rule or bill of distribution record referenced by the API.

Default Value: FND_API.G_MISS_NUM

sourcing_rule_type

Valid types must be one of the following values:

- (1) Sourcing Rule
    - (2) Bill of Distribution

Default Value: FND_API.G_MISS_NUM

return_status

Processing status of the API after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Default Value: FND_API.G_MISS_CHAR

db_flag

Indicator of the record existing in the database.

Default Value: FND_API.G_MISS_CHAR

operation

Indicator of whether the record is inserted, updated, or deleted. Valid values include:

- Create
- Update
- Delete

Default Value: FND_API.G_MISS_CHAR

See Also

Oracle Master Scheduling/MRP and Oracle Supply Chain Planning Technical Reference

Manual

Validation of Sourcing Rule Assignment API

Standard Validation

Oracle Master Scheduling/MRP validates all required columns in the Sourcing Assignment API. For specific information on the data implied by these columns, see your Oracle Master Scheduling/MRP Reference Manual for details.

If you do not want to update a particular column in:

- MRP_SRC_ASSIGNMENT_PUB.PROCESS_ASSIGNMENT

do not enter NULL for its corresponding interface parameter unless the default in the PL/SQL specification is NULL. Either use one of the missing parameter constants defined in the FND_API package (G_MISS_...), or do not pass any value at all.

For all flag parameters, pass in a Boolean constant defined in FND_API (G_TRUE or G_FALSE).

For each call, the procedure performs a check for call compatibility and then processes the assignment set and the assignment information.

Creating Assignment API Entries

When you create a new assignment, the following item level validations occur:

- Assignment_type must be Global Category, item, organization, Category-Org, Item-Org
- Sourcing_rule_id
- Assignment_set_id
- Organization_id

When you create a new assignment, the following record level validations occur:

- Assignment type attributes are conditionally required:
  - Global—sourcing rule type
  - Category—item/category, sourcing rule type
  - Item—item/category, sourcing rule type
  - Organization—organization id, customer id, ship-to -site id, sourcing rule type
  - Category-Org—organization id, customer id, ship-to -site id, item/category,

sourcing rule type

- Item-Org— organization id, customer id, ship-to -site id, item/category, sourcing rule type

When you create a new assignment, the following object level validations occur:

- Sourcing rules or bills of distribution are applicable to different assignment types:
  - Global—bill of distribution, global sourcing rule
  - Category—bill of distribution, global sourcing rule
  - Item—bill of distribution, global sourcing rule
  - Organization—local sourcing rule, global sourcing rule
  - Category-Org—local sourcing rule, global sourcing rule
  - Item-Org—local sourcing rule, global sourcing rule
- API can only assign a sourcing rule or bill of distribution to a category if there is a default value in the profile option MRP:Sourcing Rule Category Set.

If validation is successful, a record is inserted into the MRP_SR_ASSIGNMENTS table.

Updating Assignment API Entries

When you update an existing assignment, the following item level validations occur:

- Assignment_type
- Sourcing_rule_id
- Assignment_set_id
- Organization_id
- Assignment_id attribute
- If the assignment set name is changed, the new name cannot already exist in the system.
- Either assignment set name or assignment set id is required.

When you update an assignment, the following record level validation occurs:

- Assignment type attributes are conditionally required depending on assignment type. See: Creating Assignment API Entries.

- When you update an assignment, the following object level validation occurs:
- Sourcing rules or bills of distribution are applicable to different assignment types. See: Creating Assignment API Entries.

If validation is successful, a record is updated in the MRP_ASSIGNMENT_SETS table.

Deleting Assignment API Entries

When you delete an existing assignment set, the following item level validations occur:

- Assignment_id is required.
- The assignment record is deleted from MRP_SR_ASSIGNMENT table.

Error Handling

If any validation fails, the API will return error status to the calling module. The Sourcing Assignment API processes the rows and reports the following values for every record.

Condition	PROCESS_STATIS	ERROR_MESSAGE
Success	5	null
Failure	4	actual error message

See Also

Oracle Applications Message Reference Manual

6

Bills of Material Open Interfaces

This chapter covers the following topics:

- Bill of Materials API Overview
- Using the Bills of Materials Open Interfaces
- Bills of Material Entity Diagram
- Bill of Material Import Description
- Bill of Material Parameter Descriptions
- Bill of Material Package Interaction
- Bill of Material Import Error Handling and Messaging
- Bill of Material Export API
- Routing API Overview
- Routing Import Description
- Routing Parameter Descriptions
- Launching the Routing Import
- Routing Package Interaction
- Routing Import Error Handling and Messaging
- API Messaging

Bill of Materials API Overview

The Bills of Material API enables you to import your Bills of Material information from a legacy or product data management (PDM) system into Oracle Bills of Material. When you import a bill of material, you can include revision information, bill comments, components, substitute component and reference designator information in a very user friendly manner without using cryptic ID's and system specific information. The Bills of Material API can process all types of bills. The Bills of Material Object Interface insure

that your imported bills of material contain the same detail as those you enter manually in the Define Bill of Material form.

This document describes the basic business needs, major features, business object architecture and components for the Insert, Update and Delete features for the Bills of Material API.

Features

- Creating, Updating, and Deleting Bills of Material Information.

  The Bills of Material API enables you import your bills of material from external system into Oracle Bills of Material. You can update bills of material information to mimic the updates in the external system by identifying the bills of material record as an update. Similarly, when you wish to delete bill of material information, identify the record as a delete.

- Bills of Material Business Object Data Encapsulation

  When you import bills of material from an external system you are not required to provide system specific information. The Bills of material API requires business specific data that is required to define a bill.

- Synchronous Processing of information within a Bill

  The Bills of Material API processes the information within a bill synchronously. Following the business hierarchy, it processes bill header information before components.

- Asynchronous Processing of Bills of Material

  You can process multiple bills simultaneously.

- Detailed and Translatable Error Messages

  If your import fails, the Bills of Material Business API reports detailed and translatable error messages. The error message identifies the severity of the error and how it affects other records.

Using the Bills of Materials Open Interfaces

Use the following guidelines to implement the Bills of Materials open interfaces

Tables

You need to populate following interface tables with data from your legacy system: BOM_BILL_OF_MTLS_INTERFACE BOM_INVENTORY_COMPS_INTERFACE BOM_ASSY_COMMENTS_INTERFACE BOM_REF_DESGS_INTERFACE BOM_SUB_COMPS_INTERFACE| MTL_ITEM_REVISIONS_INTERFACE

Once you load the data into the interface tables, you can launch the Bill and Routing Interface program from the Import Bills and Routings form in Oracle Bills of Material or Oracle Engineering. This program assigns values, validates the data you include, and then imports the new bills of material (BOMs).

You can optionally create an item revision when you import a BOM, by inserting a value for a revision at the same time you insert your BOM data.

If you enter a value in the REVISION column of the BOM_BILL_OF_MTLS_INTERFACE table, the Bill and Routing Interface program inserts a row into the MTL_ITEM_REVISIONS_INTERFACE table. In order to assign multiple item revisions, it is better to insert data directly into the MTL_ITEM_REVISIONS_INTERFACE table.

In order to import a BOM with components, you need to populate:

BOM_BILL_OF_MTLS_INTERFACE BOM_INVENTORY_COMPS_INTERFACE

With these two tables, you can create BOM header information and assign component details.

If you want to assign standard comments, reference designators, and substitute components to your BOM, you need to populate:

BOM_ASSY_COMMENTS_INTERFACE BOM_REF_DESGS_INTERFACE BOM_SUB_COMPS_INTERFACE

PROCESS_FLAG

The column PROCESS_FLAG indicates the current state of processing for a row in the interface table. All inserted rows must have the PROCESS_FLAG set to 1.

After populating the data into the interface tables, run the Bill and Routing Interface program. The program assigns and validates all rows with a status of 1 (Pending), and then imports them into the production tables. If the assign or validate procedure fails for a row, the program sets the PROCESS_FLAG to 3 (Assign/Validation Failed) for that row.

The successful rows continue through the process of importing into the production tables. If a row fails on import, the program assigns a value of 4 (Import Failed) to the PROCESS_FLAG. Successfully imported rows have a PROCESS_FLAG value of 7 (Import Succeeded).

Transaction and request id's

The Bill and Routing Interface program automatically updates the TRANSACTION_ID and REQUEST_ID columns in each of the interface tables. The column TRANSACION_ID stores a unique id for each row in the interface table and the REQUEST_ID column stores the concurrent request id number.

Import considerations

Even though you can import bills and routings simultaneously, all routing operations must exist before you can assign a component to an operation. If a routing does not exist, you cannot assign an operation sequence to a component on a BOM.

You can simultaneously import primary and alternate BOMs. Since the Bill and Routing Interface program validates data the same way the Define Routing or Define Engineering Routing form verifies data, you cannot define an alternate bill if the primary bill does not exist. Therefore, you should import primary BOMs before importing alternate BOMs. If the program tries to validate an alternate bill before validating the primary bill, the record fails.

BOM_BILL_OF_MTLS_INTERFACE table

Required columns for BOM_BILL_OF_MTLS_INTERFACE

You must always enter values for the following required columns when you insert rows into the BOM_BILL_OF_MTLS_INTERFACE table:

ASSEMBLY_ITEM_ID

ORGANIZATION_ID

ASSEMBLY_TYPE

PROCESS_FLAG

If you create an alternate BOM, you must also enter a value in the ALTERNATE_BOM_DESIGNATOR column.

If the BOM you import references a common BOM, you must enter a value in the COMMON_ORGANIZATION_ID and COMMON_ASSEMBLY_ITEM_ID columns, or you can enter a value in the COMMON_BILL_SEQUENCE_ID column. If the bill does not reference a common bill, the Bill and Routing interface program defaults the value of the BILL_SEQUENCE_ID for the COMMON_BILL_SEQUENCE_ID.

You can specify in the ASSEMBLY_TYPE column whether the BOM is a manufacturing BOM or an engineering BOM. If you do not include a value for this column, Oracle Bills of Material defaults a value of 1 (manufacturing), and creates a manufacturing BOM. To create an engineering bill, you must enter a value of 2 (engineering) for the ASSEMBLY_TYPE column.

For each new row you insert into the BOM_BILL_OF_MTLS_INTERFACE table, you should set the PROCESS_FLAG to 1 (Pending).

Derived or defaulted values for BOM_BILL_OF_MTLS_INTERFACE

The Bill and Routing Interface program derives or defaults most of the data required to create a manufacturing or an engineering BOM. The Bill and Routing Interface program derives or defaults the columns using the same logic as the Define Bill of Material form or the Define Engineering Bill of Material form. When you populate a column in the

interface table, the program imports the row with the data you included and does not default a value.

BOM_BILL_OF_MTLS_INTERFACE	Derived or Defaulted Value
ASSEMBLY_ITEM_ID	From ITEM_NUMBER
ORGANIZATION_ID	From ORGANIZATION_CODE
LAST_UPDATE_DATE	System Date
LAST_UPDATE_BY	Userid
CREATION_DATE	System Date
CREATED_BY	Userid
COMMON_ASSEMBLY_ITEM_ID	From COMMON_ITEM_NUMBER
ASSEMBLY_TYPE	1
COMMON_BILL_SEQUENCE_ID	Sequence BOM_INVENTORY_COMPONENTS_S
COMMON_ORGANIZATION_ID	From COMMON_ORG_CODE
REQUEST_ID	From FND_CONCURRENT_REQUESTS

BOM_INVENTORY_COMPS_INTERFACE table

Required columns for BOM_INVENTORY_COMPS_INTERFACE table

Each imported record must have a value for the following columns:

PROCESS_FLAG

COMPONENT_ITEM_ID COMPONENT_SEQUENCE_ID

OPERATION_SEQ_NUM

EFFECTIVITY_DATE

BILL_SEQUENCE_ID

You must also specify a value in the ALTERNATE_BOM_DESIGNATOR column if you assign components to an alternate BOM and have not entered a value for the BILL_SEQUENCE_ID column.

When you insert rows into BOM_INVENTORY_COMPS_INTERFACE, you must set the PROCESS_FLAG to 1 (Pending) for the Bill and Routing Interface program to process the record.

Derived or defaulted column values for BOM_INVENTORY_COMPS_INTERFACE

The Bill and Routing Interface program derives or defaults most of the data required to assign components to a BOM. You can optionally include a value for derived or defaulted columns, as well as data for any of the other columns in the interface. The interface program uses the same logic to derive or default column values in the BOM_INVENTORY_COMPS_INTERFACE table as it does in the BOM_BILL_OF_MTLS_INTERFACE table. When you populate a column in the interface table, the program imports the row with the data you included and does not default a value. However, if you do not enter data in a derived or defaulted column, the program automatically imports the row with the derived or defaulted value.

BOM_BILLS_OF_MTLS_INTERFACE	Derived or Defaulted Value
COMPONENT_ITEM_ID	From COMPONENT_ITEM_NUMBER
LAST_UPDATE_DATE	System Date
LAST_UPDATE_BY	Userid
CREATION_DATE	System Date
CREATED_BY	Userid
ITEM_NUM	1
COMPONENT_QUANTITY	1
COMPONENT_YIELD_FACTOR	1
PLANNING_FACTOR	100
QUANTITY_RELATED	2
SO_BASIS	2
OPTIONAL	2
MUTUALLY_EXCLUSIVE_OPTIONS	2

INCLUDE_IN_COST_ROLLUP	1
CHECK_ATP	2
REQUIRED_TO_SHIP	2
REQUIRED_FOR_REVENUE	2
INCLUDE_ON_SHIP_DOC	2
COMPONENT_SEQUENCE_ID	Sequence, BOM_INVENTORY_COMPONENTS_S
BILL_SEQUENCE_ID	From BOM_BILL_OF_MTLS_INTERFACE or BOM_BILL_OF_MATERIALS
WIP_SUPPLY_TYPE	1
SUPPLY_LOCATOR_ID	From LOCATION_NAME
ASSEMBLY_ITEM_ID	From ASSEMBLY_ITEM_NUMBER
ORGANIZATION_ID	From ORGANIZATION_CODE
SUBSTITUTE_COMP_ID	From SUBSTITUTE_COMP_NUMBER
REQUEST_ID	From FND_CONCURRENT_REQUEST

Importing additional bill information

When you create BOMs and assign components using the Bill and Routing Interface program, you can also import additional BOM information using three different interface tables. You can import standard comments for each BOM using the BOM_ASSY_COMMENTS_INTERFACE table. You can assign component reference designators using the BOM_REF_DESGS_INTERFACE table and substitute components using the BOM_SUB_COMPS_INTERFACE table.

You can assign standard comments to any bill of material type. However, only standard components assigned to standard, model, and option class BOMs can have reference designators and substitute components.

If you insert data in the BOM_REF_DESGS_INTERFACE or BOM_SUB_COMPS_INTERFACE tables for a planning bill, the Bill and Routing Interface program fails to import the record and sets the PROCESS_FLAG to 3

(Assign/Validation Failed).

BOM_ASSY_COMMENTS_INTERFACE table

Required columns for the BOM_ASSY_COMMENTS_INTERFACE table

To import data into the BOM_ASSY_COMMENTS_INTERFACE table, you must assign a value to the following columns:

STANDARD_REMARKS_DESIGNATOR BILL_SEQUENCE_ID

PROCESS_FLAG

Derived or defaulted column values for BOM_ASSY_COMMENTS_INTERFACE

After inserting data into the BOM_ASSY_COMMENTS_INTERFACE table, the Bill and Routing Interface program derives the value for the BILL_SEQUENCE_ID column if you assign values to the columns:

ASSEMBLY_ITEM_ID

ORGANIZATION_ID ALTERNATE_BOM_DESIGNATOR

BOM_REF_DESGS_INTERFACE AND BOM_SUBS_COMPS_INTERFACE tables

Required columns for BOM_REF_DESGS_INTERFACE table

You can only import data into the BOM_REF_DESGS_INTERFACE table for standard components assigned to standard, model and option class BOMs. You must assign values to the following columns:

COMPONENT_REFERENCE_DESIGNATOR COMPONENT_SEQUENCE_ID

PROCESS_FLAG

Required columns for BOM_SUBS_COMPS_INTERFACE table

You can only import data into the BOM_SUBS_COMPS_INTERFACE table for standard components assigned to standard, model and option class BOMs. You must assign values to the following columns:

SUBSTITUTE_COMPONENT_ID SUBSTITUTE_ITEM_QUANTITY COMPONENT_SEQUENCE_ID

PROCESS_FLAG

Derived or default values for BOM_REF_DESGS_INTERFACE and BOM_SUB_COMPS_INTERFACE

After inserting data into the BOM_REF_DESGS_INTERFACE or BOM_SUB_COMPS_INTERFACE tables, the Bill and Routing Interface program derives the value for the BILL_SEQUENCE_ID column if you assign values to the following columns:

ASSEMBLY_ITEM_ID

ORGANIZATION_ID ALTERNATE_BOM_DESIGNATOR

The program also assigns a value for the COMPONENT_SEQUENCE_ID column if you enter values into the following columns:

BILL_SEQUENCE_ID

COMPONENT_ITEM_ID

OPERATION_SEQ_NUM

EFFECTIIVITY_DATE

Validating Interface table rows

After you load the BOM and component data, the Bill and Routing Interface program validates the required data for the six interface tables. BOM validation insures that each row has an included or defaulted value for all the required columns and verifies the same way as the Define Bill of Material form and the Define Engineering Bill of Material form validate manually entered bills. For example, you cannot import a standard bill and assign model, option class or planning items as components.

If the Bill and Routing Interface program cannot assign a value to a row or validate that row, the program sets the PROCESS_FLAG to 3 (Assign/Validation Failed) and inserts a row in the MTL_INTERFACE_ERRORS table.

To identify the error message for a failed row, the program automatically populates the UNIQUE_ID column in the error interface table with the same value as the TRANSACTION_ID value. Each error has a value for the MESSAGE_NAME and REQUEST_ID columns in the error interface table. The MESSAGE_NAME column corresponds to messages stored in the Oracle Application Message Dictionary. The REQUEST_ID column stores the concurrent request id. If the program detects any internal database error, the program stores the internal error in the MESSAGE_NAME column and stores the specific database error message in the ERROR_MESSAGE column.

If you import a BOM with multiple components and one of the components fails validation, then the bill will be created without the failed component. If, however, the row in the BOM_BILL_OF_MTLS_INTERFACE table fails, the BOM and and all of its details are not imported.

Correcting failed rows

You can review and report rows in the interface tables using SQL*Plus or any custom report you develop. Since all rows in the interface table have a value for PROCESS_FLAG, you can easily identify records that are successfully imported into Oracle Bill of Material and Oracle Engineering, or records that failed validation or import. You can also identify individual records by the unique value for the TRANSACTION_ID column.

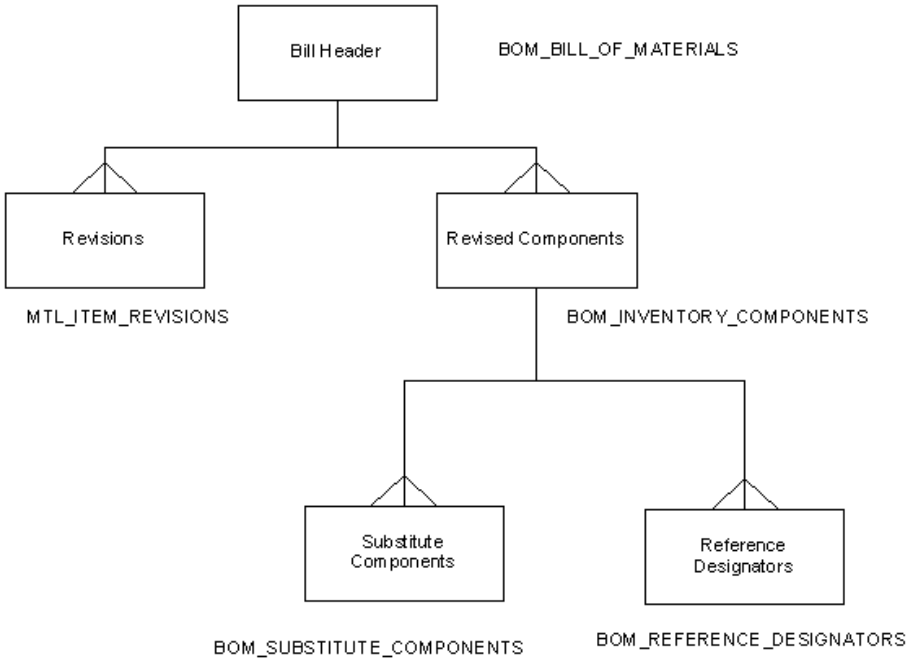
You can update any row from the interface tables using SQL*Plus. If you update a row

to resolve invalid data, you must set the PROCESS_FLAG to 1 (Pending) for that row.

If you delete a failed row and insert a replacement row, you should set the PROCESS_FLAG to 1 (Pending) for the new row. When you resubmit the Bill and Routing Interface program, all rows pending validation are processed.

Bills of Material Entity Diagram

The following diagram shows the table structure for the ECO business object along with all its entities:



Bills of Material Business Object Architecture

The Bills of Material Business Object Architecture is based on the hierarchical definition of BOM of material in Oracle Bills of Material. To use the Bills of Material Business object interface you only need to know structure of your BOM. As in a genealogical tree, the entity at the top is the parent. The entities connected directly below are its children.

Bills of Material Header

The BOM header entity is the topmost entity in the Bills of Material hierarchy. You can process more than one header record at a time. The header entity will contain information such as the header item name, organization in which the item exists, alternate designator if bill is an alternate and the revision.

Revisions

The revisions entity enables you to define any number of revisions for a header item. When you create or update a bill of material you can choose to create a new revision or modify an existing revision. You can also define different versions of a bills of material within the same revision. To identify a revision you must specify the organization in which exists, assembly item number and revision.

Components

A component cannot exist without a header entity. To identify a component you must specify the header information such as the item number, organization, alternate designator (if you are adding the component to an alternate bill), effective date, item number, and operation sequence number.

Substitute Component

A substitute component cannot exist without a component. To identify a substitute component, you must identify the following information: organization, assembly item number, effectivity date, alternate if the component belongs on an alternate bill, component item number, operation sequence number, and substitute component number.

Reference Designator

A reference designator cannot exist without a component. To identify a reference designator, you must identify the following information: organization, assembly item number, effectivity date, alternate if the component belongs on an alternate bill, component item number, operation sequence number, and substitute component number.

Production Tables

Each of the entities shown in the Bills of Material entity diagram maps to a corresponding table in the database. The following are the existing production tables that exist, and the information they store.

Production Table	Description
BOM_BILLS_OF_MATERIAL	Stores information about the Bills of Material header item or the Assembly Item

Production Table	Description
MTL_ITEM_REVISIONS	Stores information about Bill of Materials revisions
BOM_INVENTORY_COMPONENTS	Stores information about the un-implemented single-level BOM components
BOM_REFERENCE_DESIGNATORS	Stores information about the un-implemented BOM component reference designators
BOM_SUBSTITUTE_COMPONENTS	Stores information about the un-implemented substitute components associated with a BOM component
BOM_EXPORT_TAB	Stores exploded bill information for the specified assembly for all subordinate organizations in a specified organization hierarchy.

Business Logic and Business Object Rules

Business logic helps model the business. It includes all constraints and considerations that the business needs to maintain and process data successfully. This implies that certain rules must be imposed on incoming data (business objects) to ensure its validity within the context of the business.

Bill of Material Import Description

Step	Purpose	Description	Error
Step 1: Pass Business Object to Public API	The program will try to import it	Caller must pass one business object at a time. There should be only one Header record. There may be more than one record for other entities.	-N/A-

Step	Purpose	Description	Error
Step 2: Check for Organization uniformity	We must ensure that all records in a business object belong to the same Organization.	Derive Organization_Id from Organization_Code Store Organization_Id value in System_Information record.	Severe Error I
Step 3: Save system information	Saves system-specific information in System_Information record since it is common to the whole business object. This information is stored in the database along with the record	Initialize User_Id, Login_Id, Prog_Appid, Prog_Id in System_Information record. Pull in values of profiles ENG: Standard Item Access, ENG: Model Item Access and ENG: Planning Item Access into STD_Item_Access, MDL_Item_Access and PLN_Item_Access respectively.	Quit import of business object
Step 4: Pick up highest level un-processed record	The import program processes a parent and all its direct and indirect children, before moving onto to a sibling. So, the highest level parent must be chosen.	The highest level record with a {return status = NULL} is picked. When there are no more records, the program exits and transfers control to the caller.	-N/A-

Step	Purpose	Description	Error
Step 5: Convert user-unique index to unique index	Unique index helps uniquely identify a record in the database, and may consist of more than one column. User-unique index is a user-friendly equivalent of the unique index. It serves the following purposes: The user need not enter cryptic Ids If a user unique index could not be derived for a parent, it's entire lineage is error-ed out since members of the lineage are referencing an invalid parent.	Derive unique index columns from user-unique index columns.	Severe Error III
Step 6: Existence Verification	The record being updated or deleted in the database must already exist. But a record being created must not. Such an error in a record must cause all children to error out, since they are referencing an invalid parent.	For CREATE, the record must not already exist. For UPDATE and DELETE, the record must exist. Query up database record into the associated OLD record.	Severe Error III

Step	Purpose	Description	Error
Step 7: Check Lineage	We must ensure that the linkage of records is correct in the business object. That is, child records must reference valid parents. A valid parent is one that exists, and is truly the current record's parent in the database.	Perform lineage checks for entity records that do not belong to the top-most entity in the hierarchy, based on Transaction_Type and the following factors: Immediate parent being referenced exists in the database, and, for UPDATE and DELETE, is truly the parent of this record in the database, OR If there is no immediate parent record in the business object, the indirect parent being referenced exists and is really the parent of the current record's parent in the database.	Severe Error III
Step 8(a): Check operability of parent items and current item (if applicable)	A assembly item and any of it's components cannot be operated upon if the assembly item is implemented or canceled.	Check if System_Information record has this information. If not, find it in the database assembly item record, and set System_Information flags accordingly.	Fatal Error III or Fatal Error II (depending on affected entity)

Step	Purpose	Description	Error
Step 8(b): Check operability of parent items and current item (if applicable)	A assembly item and any of it's components cannot be operated upon if the user does not have access to the assembly item type.	Compare assembly item BOM_Item_Type against the assembly item access fields in the System_Information record.	Fatal Error III or Fatal Error II (depending on affected entity)
Step 9: Value-Id conversions	There are user-friendly value columns that derive certain Id columns. The value columns free up the user from having to enter cryptic Ids, since the Ids can be derived from them.	Derive Ids from user-friendly values	CREATE: Severe Error IV. Others: Standard Error
Step 10: Required Fields checking	Some fields are required for an operation to be performed. Without them, the operation cannot go through. The user must enter values for these fields.	Check that the required field columns are not NULL.	CREATE: Severe Error IV. Other: Standard Error
Step 11: Attribute validation (CREATEs and UPDATEs)	Each of the attributes/fields must be checked individually for validity. Examples of these checks are: range checks, checks against lookups etc.	Check that user-entered attributes are valid. Check each attribute independently of the others	CREATE: Severe Error IV. UPDATE: Standard Error

Step	Purpose	Description	Error
Step 12: Populate NULL columns (UPDATEs and DELETEs)	The user may send in a record with certain values set to NULL. Values for all such columns are copied over from the OLD record. This feature enables the user to enter minimal information for the operation.	For all NULL columns found in business object record, copy over values from OLD record.	-N/A-
Step 13: Default values for NULL attributes (CREATEs)	For CREATEs, there is no OLD record. So the program must default in individual attribute values, independently of each other. This feature enables the user to enter minimal information for the operation to go through.	For all NULL columns found in business object record, try to default in values, either by retrieving them from the database, or by having the program assign values.	Severe Error IV
Step 14: Check conditionally required attributes	Some attributes are required based on certain external factors such as the Transaction_Type value.	Perform checks to confirm all conditionally required attributes are present.	Severe Error IV
Step 15: Entity level defaulting	Certain column values may depend on profile options, other columns in the same table, columns in other tables, etc. Defaulting for these columns happens here.	For all NULL columns in record, try to default in values based on other values. Set all MISSING column values to NULL.	CREATE: Severe Error IV. Update: Standard Error

Step	Purpose	Description	Error
Step 16: Entity level validation	This is where the whole record is checked. The following are checked: Non-updateable columns (UPDATEs): Certain columns must not be changed by the user when updating the record. Cross-attribute checking: The validity of attributes may be checked, based on factors external to it. Business logic: The record must comply with business logic rules.	Perform checks against record in the order specified in the -Purpose- column.	CREATE: Severe Error IV. UPDATE: Standard Error
Step 17: Database writes	Write record to database table.	Perform database write: Insert record for CREATE Overwrite record for UPDATE and CANCEL Remove record for DELETE	-N/A-

Step	Purpose	Description	Error
Step 18: Process direct and indirect children	The programwill finish processing an entire branch before moving on to a sibling branch. A branch within the business object tree consists of all direct and indirect children.	Pick up the first un-processed child record and Go to Step 5. Continue until all direct children have been processed. Then pick up the first un-processed indirect child record and do the same as above. When no more records are found, Go to Step 20.	-N/A-
Step 19: Process siblings	When an entire branch of a record has been processed, the siblings of the current record are processed. The sibling may also contain a branch. So the processing for the sibling will be exactly the same as the current record.	Pick up the first un-processed sibling record and Go to Step 5. Continue through the loop until all siblings have been processed. When no more records are found, Go to Step 21.	-N/A-
Step 20: Process parent record's siblings	Once all the siblings have been processed, the program will move up to the parent (of this entire branch) and process all of its siblings (which will contains branches of their own).	Go up to parent and pick up the first un-processed sibling of the parent. Go to Step 5. Continue through the loop until all siblings have been processed. When there are no more records, Go to Step 4.	-N/A-

Bill of Material Parameter Descriptions

Bills of Material Exposed Columns

BOM Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	-	x	-
Organization_Code	VARCHAR2(3)	-	x	-
Alternate_BOM_Code	VARCHAR2(10)	-	x	-
Common_Assembly_Name	VARCHAR2(81)	-	-	X
Assembly_Type	NUMBER	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Original_System_Reference	VARCHAR2(50)	x	-	-
Return_Status	VARCHAR2(1)	-	x	-
Assembly_Type	NUMBER	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Attribute Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x

BOM Name	Type	Required	Derived	Optional
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x

Organization_Code Organization Identifier.

Default Value: FND_API.G_MISS_CHAR

Assembly_Item_Name Inventory item identifier of manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Alternate_Bom_Designator Alternate designator code.

Default Value: FND_API.G_MISS_CHAR

Change_Description Change description.

Default Value: FND_API.G_MISS_CHAR

Common_Assembly_Item_Name Inventory item identifier of common assembly.

Default Value: FND_API.G_MISS_CHAR

Common_Organization_Code Organization identifier of a common bill.

Default Value: FND_API.G_MISS_CHAR

Assembly_Type Type of assembly.

1. Engineering Bill
2. Manufacturing Bill

Default Value: FND_API.G_MISS_NUM

Attribute_Category Descriptive flexfield structure defining column.

Default Value: FND_API.G_MISS_CHAR

Attribute1-15 Descriptive flexfield segment.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Transaction_Type Type of action: Create, Update, or delete.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

Bills of Material Components Exposed Columns

BOM Name	Type	Required	Derived	Optional
Organization_Code	VARCHAR2(3)	-	-	x
Assembly_Item_Name	VARCHAR2(81)	-	-	x
Alternate_Bom_Code	VARCHAR2(10)	-	-	x
Start_Effective_Date	DATE	-	-	x
Operation_Sequence_Number	NUMBER	-	-	x
Component_Item_Name	VARCHAR2(81)	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Organization_Code	VARCHAR2(3)	-	-	x
Assembly_Item_Name	VARCHAR2(81)	-	-	x
Disable_Date	DATE	-	-	x
New_Effectivity_Date	DATE	-	-	x
New_Operation_Sequence_Number	NUMBER	-	-	x
Item_Sequence_Number	NUMBER	-	-	x
Quantity_Per_Assembly	NUMBER	-	-	x

BOM Name	Type	Required	Derived	Optional
Planning_Percent	NUMBER	-	-	x
Projected_Yield	NUMBER	-	-	x
Include_In_Cost_Rollup	NUMBER	-	-	
WIP_Supply_Type	NUMBER	-	-	x
Supply_Subinventory	VARCHAR2(10)	-	-	x
Locator_Name	VARCHAR2(81)	-	-	x
SO_Basis	NUMBER	-	-	x
Optional	NUMBER	-	-	x
Mutally_Exclusive	NUMBER	-	-	x
Check_ATP	NUMBER	-	-	x
Minimum_Allowed Quantity	NUMBER	-	-	x
Maximum_Allowed_Quantity	NUMBER	-	-	x
Attribute_Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x

BOM Name	Type	Required	Derived	Optional
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x
From_End_Item_Unit_Number	VARCHAR2(30)	-	-	x
To_End_Item_Unit_Number	VARCHAR2(30)	-	-	x

Organization_Code Organization Identifier.

Default Value: FND_API.G_MISS_CHAR

Assembly_Item_Name Inventory item identifier of manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Start_Effective_Date Effective Date.

Default Value: FND_API.G_MISS_DATE

Disable_Date Disable Date

Default Value: FND_API.G_MISS_DATE

Operation_Sequence_Number Routing Operation unique identifier.

Default Value: FND_API.G_MISS_NUM

Component_Item_Name Component Item identifier.

Default Value: FND_API.G_MISS_CHAR

Alternate_BOM_Code Alternate BOM designator code.

Default Value: FND_API.G_MISS_CHAR

New_Effectivity_Date New effective date.

Default Value: FND_API.G_MISS_DATE

New_Operation_Sequence_Number New operation sequence number.

Default Value: FND_API.G_MISS_NUM

Item_Sequence_Number Item unique identifier.

Default Values: FND_API.G_MISS_NUM

Quantity_Per_Assembly Quantity per assembly.

FND_API.G_MISS_NUM

Planning_Percent Planning Percentage.

Default Value: FND_API.G_MISS_NUM

Projected_Yield Projected yield for an operation.

Default Value: FND_API.G_MISS_NUM

Include_In_Cost_Rollup Flag that indicates to include the material cost of a component in the standard cost of the assembly.

1. Yes
2. No

Default Value: FND_API.G_MISS_NUM

Wip_Supply_Type WIP supply type code.

Default Value: FND_API.G_MISS_NUM

So_Basis Quantity basis Oracle Order Management uses to determine how many units of a component to put on an order.

1. Yes
2. No

Default Value: FND_API.G_MISS_NUM

Optional Flag indicating if a flag is optional in a bill.

1. Yes
2. No

Default Value: FND_API.G_MISS_NUM

Mutually_Exclusive Flag indicating if one or more children of a component can be picked when taking an order.

1. Yes
2. No

Default Value: FND_API.G_MISS_NUM

Check_Atp Flag indicating if ATP check is required.

1. Yes
2. No

Default Value: FND_API.G_MISS_NUM

Shipping_Allowed Flag indicating if a component may be shipped.

Default Value: FND_API.G_MISS_NUM

Required_To_Ship Flag indicating that you must include this component of a pick-to-order item in the first shipment of the pick to order item.

FND_API.G_MISS_NUM

Required_For_Revenue Flag that prevents invoicing for the pick-to-order item until you ship the component.

Default Value: FND_API.G_MISS_NUM

Include_On_Ship_Docs Flag indicating if the component of the configure-to-order item should appear on Order Entry documents.

Default Value: FND_API.G_MISS_NUM

Quantity_Related Identifier to indicate if this component has quantity related reference designators.

Default Value: FND_API.G_MISS_NUM

Supply Subinventory Supply subinventory.

Default Value: FND_API.G_MISS_CHAR

Location_Name Supply Location Name.

Default Value: FND_API.G_MISS_CHAR

Minimum_Allowed_Quantity Minimum quantity allowed on an order.

Default Value:FND_API.G_MISS_NUM

Maximum_Allowed_Quantity Maximum quantity allowed for an order.

Default Value: FND_API.G_MISS_NUM

Component_Remarks Component Remarks.

Default Value: FND_API.G_MISS_CHAR

Attribute_Category Descriptive flexfield structure defining category

Default Value: FND_API.G_MISS_CHAR

Attributes 1-15 Descriptive Flexfield segments.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

From_End_Item_Unit_Number From end item unit number.

Default Value: FND_API.G_MISS_CHAR

To_End_Item_Unit_Number To end item unit number.

FND_API.G_MISS_CHAR

Item Revision Exposed Columns

Item Revision	Type	Required	Derived	Optional
Organization_Code	VARCHAR2(3)	-	-	x
Assembly_Item_Name	VARCHAR2(81)	-	-	x
Start_Effective_Date	VARCHAR2(3)	-	-	x
Revision	VARCHAR2(15)	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Revision_Comment	VARCHAR2(240)	-	-	x
Return_Status	VARCHAR2(1)	-	-	x
Attribute Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x

Item Revision	Type	Required	Derived	Optional
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x

Organization_Code Organization Identifier.

Default Value: FND_API.G_MISS_CHAR

Assembly_Item_Name Inventory item identifier for manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Start_Effective_Date Effective date.

Default Value: FND_API.G_MISS_DATE

Operation_Sequence_Number Routing operation unique identifier.

Default Value: FND_API.G_MISS_NUM

Component_Item_Name Component Item identifier.

Default Value: FND_API.G_MISS_CHAR

Alternate_Bom_Code Alternate designator code.

Default Value: FND_API.G_MISS_CHAR

Reference_Designator_Name Component reference designator.

Default Value: FND_API.G_MISS_CHAR

Ref_Designator_Comment Reference designator comment.

Default Value: FND_API.G_MISS_CHAR

Attribute_Category Descriptive flexfield structure defining category.

Default Value: FND_API.G_MISS_CHAR

Attributes 1-15 Descriptive Flexfield segments.

Default Value: FND_API.G_MISS_CHAR

New_Reference_Designator Updated value for the old reference designator.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

Substitute Component Exposed Columns

Substitute Component	Type	Required	Derived	Optional
Organization_Code	VARCHAR2(3)	-	-	x
Assembly_Item_Name	VARCHAR2(81)	-	-	x
Start_Effective_Date	VARCHAR2(3)	-	-	x
Operation_Sequence_Number	NUMBER	-	-	x

Substitute Component	Type	Required	Derived	Optional
Alternate_BOM_Code	VARCHAR2(10)	-	-	x
Component_Item_Name	VARCHAR2(81)	-	-	x
Substitute_Component_Name	VARCHAR2(81)	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Substitute_Item_Quantity	Number	-	-	x
Return_Status	VARCHAR2(1)	-	-	x
Attribute Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	Attribute3	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x

Substitute Component	Type	Required	Derived	Optional
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x

Organization_Code Organization Identifier.

Default Value: FND_API.G_MISS_CHAR

Assembly_Item_Name Inventory item identifier for manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Start_Effective_Date Component Effective Date.

FND_API.G_MISS_DATE

Operation_Sequence_Number The Routing step in which you use the component.

FND_API.G_MISS_NUM

Component_Item_Name The Component Item Name.

FND_API.G_MISS_CHAR

Alternate_BOM_Code Alternate BOM identifier.

FND_API.G_MISS_CHAR

Substitute_Component_Name The name of the substitute component.

FND_API.G_MISS_CHAR

Substitute_Item_Quantity The usage quantity of the substitute component.

FND_API.G_MISS_NUM

Attribute_Category Descriptive flexfield structure defining category.

Default Value: FND_API.G_MISS_CHAR

Attributes 1-15 Descriptive flexfield segments.

Default Value: FND_API.G_MISS_CHAR

New_Reference_Designator Updated value for the old reference designator.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current item.

Launching the Import

The following details explain launching the import:

The Three Step Rule:

In order to use the business object APIs effectively, the user must follow the three step rule:

1. Initialize the system information.
2. Call the Public API.
3. Review all relevant information after the Public API has completed.

Step 1: Initialize the system information

Each database table requires you to enter system information, such as who is trying to update the current record. You initialize certain variables to provide the information to the import. The program retrieves system information from these variables during operation. To initialize the variables, the you must call the following procedure:

FND_GLOBAL.apps_initialize

user_idIN NUMBER

resp_idIN NUMBER

resp_appl_idIN NUMBER

security_group_idIN NUMBER DEFAULT 0

Pointers

1. This procedure initializes the global security context for each database session.
2. This initialization should be done when the session is established outside of a normal forms or concurrent program connection.
3. User_id is the FND User Id of the person launching this program.
4. Resp id is the FND Responsibility Id the person is using.
5. Resp_appl_id is the Application Responsibility Id.
6. Security_group_id is the FND Security Group Id.

Step 2: Call the Public API

You must call the public API programmatically, while sending in one record at a time. The Public API returns the processed record, the record status, and a count of all associated error and warning messages.

The procedure to call is Process_Eco, and the following is its specification:

PROCEDURE Process_BOM

(p_api_version_number IN NUMBER

, p_init_msg_list IN VARCHAR2:= FND_API.G_FALSE

, x_return_status OUT VARCHAR2

, x_msg_count OUT NUMBER

, x_msg_data OUT VARCHAR2

, p_ECO_rec IN Eco_Rec_Type:= G_MISS_ECO_REC

, p_assembly_item_tbl IN Revised_Item_Tbl_Type:=

G_MISS_REVISED_ITEM_TBL

, p_rev_component_tbl IN Rev_Component_Tbl_Type:=

G_MISS_REV_COMPONENT_TBL

, p_ref_designator_tbl IN Ref_Designator_Tbl_Type:=

G_MISS_REF_DESIGNATOR_TBL

, p_sub_component_tbl IN Sub_Component_Tbl_Type:=

G_MISS_SUB_COMPONENT_TBL

, x_assembly_item_tbl OUT Revised_Item_Tbl_Type

, x_rev_component_tbl OUT Rev_Component_Tbl_Type

, x_ref_designator_tbl OUT Ref_Designator_Tbl_Type

, x_sub_component_tbl OUT Sub_Component_Tbl_Type)

As is obvious from the above specification, all IN parameters begin with *p_*. All OUT parameters begin with *x_*. The following is a description of these parameters:

- p_api_version_number: This parameter is required. It is used by the API to compare the version numbers of incoming calls to its current version number, and return an unexpected error if they are incompatible. See section 4.1 of the Business Object Coding Standards document for a detailed description of this parameter.

- p_init_msg_list: This parameter, if set to TRUE, allows callers to request that the API do the initialization of the message list on their behalf. On the other hand, the caller may set this to FALSE (or accept the default value) in order to do the initialization itself by calling Error_Handler.Initialize.

- p_assembly_item_tbl, p_rev_component_tbl, p_ref_designator_tbl, p_sub_component_tbl: This is the set of data structures that represents the incoming business object. p_assembly_item is a record that holds the Bill of Materials header for a BOM. All the other data structures are PL/SQL tables of records that hold records for each of the other entities. All these data structures directly correspond to the entities shown in the BOM entity diagram.

Please note that any of these data structures may be sent in empty (set to NULL) to indicate that there are no instances of that entity in the business object being sent in.

- x_assembly_item_tbl, x_rev_component_tbl, x_ref_designator_tbl, x_sub_component_tbl: This is the set of data structures that represents the outgoing business object. These records essentially constitute the whole business object as it was sent in, except now it has all the changes that the import program made to it through all the steps in the Process Flow. These records can be committed, rolled back, or subjected to further processing by the caller. All these data structures directly correspond to the entities shown in the BOM entity diagram.

- x_return_status: This is a flag that indicates the state of the whole business object after the import. If there has been an error in a record, this status will indicate the fact that there has been an error in the business object. Similarly, if the business object import has been successful, this flag will carry a status to indicate that. The caller may look up this flag to choose an appropriate course of action (commit, rollback, or further processing by the caller). The following is a list of all the possible business object states:

CODE	MEANING
'S'	Success
'E'	Error
'F'	Fatal Error
'U'	Unexpected Error

- x_msg_count: This holds the number of messages in the API message stack after the import. This parameter returns a 0 when there are no messages to return.
- x_msg_data: This returns a single message when the message count is 1. The purpose of this parameter is to save the caller the extra effort of retrieving a lone message from the message list. This parameter returns NULL when there is more than one message.

As mentioned above, the Public API must be called programmatically. The caller here may be a form, a shell, or some other API which serves to extend the functionality of the import program.

> **Note:** A record must have an error status of NULL for it to be processed. If it has any other value, it will not be picked up for processing. The user must remember to NULL out this field when sending in a record.

Step 3: Review all relevant information after the Public API has completed

You must look up:

- All error status that the Public API returns, including, the overall business object error status, as well as the individual record statuses.
- All record attributes to see what change occurred after applying business logic to these records.
- All error and warning messages in the API message list.

You can access the API message list using the following procedures and functions in the Error_Handler package:

1. Initializing the message list: The following procedure clears the message list and initializes all associated variables

2. PROCEDURE Initialize;

3. Go to the start of the list: The following procedure reset the message index to the start of the list so the user can start reading from the start of the list

   PROCEDURE Reset;

4. Retrieving the entire message list: The following procedure will return the entire message list to the user

   PROCEDURE Get_Message_List

   (x_message_list OUT Eng_Eco_Pub.Error_Tbl_Type);

5. Retrieving messages by entity: One implementation of procedure Get_Entity_Message will return all messages pertaining to a particular entity (p_entity_id), denoted by the symbols ECO (ECO Header), RI (Revised Item), RC (Revised Component), SC (Substitute Component), RD (Reference Designator).

   PROCEDURE Get_Entity_Message

   (p_entity_id IN VARCHAR2

   , x_message_list OUT Eng_Eco_Pub.Error_Tbl_Type

   );

6. Retrieving a specific message: Another implementation of procedure Get_Entity_Message will return the message pertaining to a particular entity (p_entity_id), at a specific array index in that entity table. The entity is denoted by the symbols BH (BOM Header), RC (Revised Component), SC (Substitute Component), RD (Reference Designator). The entity index (p_entity_index) is the index in the entity array.

   PROCEDURE Get_Entity_Message

   (p_entity_id IN VARCHAR2

   , p_entity_index IN NUMBER

   , x_message_text OUT VARCHAR2

   );

7. Retrieving the current message: Procedure Get_Message will returns the message at the current message index and will advance the pointer to the next index. If the user tries to retrieve beyond the size of the message list, then the message index will be reset to the beginning of the list.

   PROCEDURE Get_Message

   (x_message_text OUT VARCHAR2

   , x_entity_index OUT NUMBER

, x_entity_id OUT VARCHAR2

);

8. Deleting a specific message: Procedure Delete_Message enables the user to delete a specific message at a specified entity index (p_entity_index) within the PL/SQL table of a specified entity (p_entity_id). The entity is denoted by the symbols BH (Bills Header), RC (Revised Component), SC (Substitute Component), RD (Reference Designator). The entity index (p_entity_index) is the index in the entity array.

   PROCEDURE Delete_Message

   (p_entity_id IN VARCHAR2

   , p_entity_index IN NUMBER);

9. Deleting all messages for a certain entity: Another implementation of procedure Delete_Message lets the user delete all messages for a particular entity (p_entity_id). The entity is denoted by the symbols BH (Bill Header), RC (Revised Component), SC (Substitute Component), RD (Reference Designator).

   PROCEDURE Delete_Message

   (p_entity_id IN VARCHAR2);

10. Get a count of all messages: The following functions returns the total number of messages currently in the message list

    FUNCTION Get_Message_Count RETURN NUMBER;

11. Dumping the message list: The following message generates a dump of the message list using dbms_output

    PROCEDURE Dump_Message_List;

Bill of Material Package Interaction

The Public Package - BOM_BO_PUB

This packages allows only one business object through at a time. All records in a specific business object must belong to the business object. For example, the records associated with an engineering change order would make up an instance of the ECO business object. In this example all of the records in the ECO business object instance must reference the same engineering change order.

Main Procedure: Process_BOM

1. Set business object status to S.

2. Check that all records in the business object belong to the same Bill, i.e., all records must have the same Assembly Item Name and Organization_Code combination.

Description	Cause of Failure	Error	Message
If there is an Bill Header in the business object, check that all records have the same Assembly_item_name and Organization_Code values as the Header. If the business object does not have an Bill Header record, check that all records have the same Assembly_Item_Name and Organization_Code combination as the first highest level entity record picked up.	Any records have mismatched Assembly_Item_Name and Organization_Code values	Severe Error I	BOM_MUST_BE_IN_SAME_BOM: All records in a business object must belong to the same Bill of Material. That is, they must all have the same Assembly Name and organization. Please check your records for this.

3. Derive Organization_Id from Organization_Code and copy this value into all business object records.

Column	Description	Error	Message
Organization_Id	Derive using Organization_Code from table MTL_PARAMTERS	Severe Error I	BOM_ORG_INVALID: The Organization <org_id> you entered is invalid.

Unexpected Error Other Message:

BOM_UNEXP_ORG_INVALID: This record was not processed since an unexpected error while performing a value to id conversion for organization code.

1. Pass business object into Private API if the business object status is still set to S. Also pass the Assembly Item Name and Organization_Id to Private API, to identify this business object instance.

2. Accept processed business object and return status from Private API after the import, and pass it back to the calling program.

Bill of Material Import Error Handling and Messaging

Error Handling Concepts

Error handling depends on the severity of the error, the scope of the error, and how the error affects the lineage (child record error states). When an error occurs, records are marked so that erroneous records are not processed again.

Error Severity Levels

Severity levels help distinguish between different types of errors since the import program behaves differently for each of these errors. The following is a list of the error severity levels recognized by the import program

CODE	MEANING
'W'	Warning / Debug
'E'	Standard Error
'E'	Severe Error
'F'	Fatal Error
'U'	Unexpected error

Error States

Error states serve two purposes:

- They convey to the user the exact type of error in the record.
- They help the import program identify the records that do not need to be processed .

CODE	MEANING
'S'	Success
'E'	Error
'F'	Fatal Error
'U'	Unexpected Error
'N'	Not Processed

Error Scope

This indicates what the depth of the error is in the business object, that is, how many other records in the business object hierarchy the current error affects.

CODE	MEANING
'R'	Error affects current 'R'ecord
'S'	Error affects all 'S'ibling and child records
'C'	Error affects 'C'hild records
'A'	Error affects 'A'll records in business object

Child Error States

If an error in a record affects child records, the status of the child may vary based on the type of error. There are two error states that indicate how the child is affected:

CODE	MEANING
'E'	Error
'N'	Not Processed

Error Classes

There are three major classes that determine the severity of the problem.

Expected errors: These are errors the program specifically looks for in the business object, before committing it to the production tables.

Standard Error: This error causes only the current record to be error-ed out, but is not serious enough to affect any other records in the object. The current record status is set to E. For example: Bill of Material entry already exists.

1. Severe Error I: This error affects all records. All record statuses are set to E. This error is usually a organization uniformity error. All records must have the same organization code.

2. Severe Error II: This error affects all records that are children of this record's parent, when the parent is not in the business object. A characteristic of this record's parent caused the error, so all it's siblings and their children also get a record status of E. This error usually occurs when a lineage check fails.

3. Severe Error III: This error not only affects the current record but also its child records in the business object. The child record statuses are set to E. Please check your child records for errors as well as the current record. This error is usually a user-unique to unique index conversion error.

4. Severe Error IV: This error affects the current record and its child records since the program cannot properly process the child records. The child record statuses are set to N. This type of errors occur when there are errors on CREATEs.

5. Fatal Error I: These errors occur when it is not possible to perform any operation on the ECO. Such errors affect the entire business object. All record statuses are set to F. The following are situations that cause this error:

6. You do not have access to this Item Type.

7. Fatal Error II: This error affects all records that are children of this record's parent, when the parent is not in the business object. A characteristic of this record's parent caused the error, so all it's siblings and their children also get a record status of F. This usually occurs when the user tries to create, update, or delete a component of a

assembly item that the user does not have access to.

8. Fatal Error III: These errors affects the current record and its children, since it is not possible to perform any operation on these records. The current record and all its child record statuses are set to F. The following situations cause this error:
   - You do not have access to the component item's BOM item type

Unexpected errors

All errors that are not expected errors are unexpected errors. These are errors that the program is not specifically looking for, for example, the user somehow loses the database connection.

Warnings: These are messages for information only. The purpose of warnings is:

1. to warn the user of problems that may occur when the bill is used by other manufacturing applications. For example: WIP Supply Type must be Phantom.
2. to inform the user of any side-effects caused by user-entered data.

Child record:

All records carry the unique keys for all parents above them. A child record (of a particular parent) is one that holds the same values for the unique key columns as the parent record.

Sibling record

A sibling record (of the current record) is one that holds the same parent unique key column values as the current record. For example, a component record that holds the same parent assembly item unique key column values as the current component record, is a sibling of the current component record. Likewise, a reference designator record that holds the same parent component unique key column values as a substitute component is a sibling of the substitute component.

Business Object Error Status

This indicates the state of the whole business object after the import. As soon as the import program encounters an erroneous record, it sets the business object error status (also called return status) appropriately to convey this to the user. It is then up to the user to locate the offending record(s) using the individual record error statuses as indicated below. The caller may also use the business object return status to choose an appropriate course of action (commit, rollback, or further processing by the caller).

The following is a list of all the possible business object states:

CODE	MEANING
'S'	Success
'E'	Error
'F'	Fatal Error
'U'	Unexpected Error

Record Error Status

This is the state of the record after the import (success or error). The error status is also referred to as return status or error state in this document. The error status helps locate erroneous records in a business object that has error-ed out. The following are important pointers about the record error status.

- Every record is assigned an error status by the import program. Hence, if a record has a NULL return status, it means that the import program has not gotten to it yet.
- The user must send in records with {return status = NULL}. The import program will not look at records that already have an error status value, since it assumes that a record with an error status value has already been looked at by the program.

The following shows how the error status, error scope, and child statuses relate together to constitute the different error classes for records:

Error	Status	Scope	Child Statuses
Warning	S: Success	R: Record Only	-N/A-
Standard Error	E: Error	R: Record Only	-N/A-
Severe Error I	E: Error	A: All Records	E: Error
Severe Error II	E: Error	S: Current, Sibling and Child Records	E: Error
Severe Error III	E: Error	C: Current and Child Record	E: Error

Error	Status	Scope	Child Statuses
Severe Error IV	E: Error	C: Current and Child Records	N: Not Processed
Fatal Error I	F: Fatal Error	A: All Records	-N/A
Fatal Error II	F: Fatal Error	S: Current, Sibling and Child Records	-N/A
Fatal Error III	F: Fatal Error	C: Current and Child Record	-N/A
Unexpected Error	U: Unexpected Error	-N/A-	N: Not Processed

API Messaging

Error Message Table

All messages are logged in the Error Message Table. This is a PL/SQL table (array) of messages. Please see Accessing Messages in the Launching the Import section of this document on how to access these messages.

The following is a description of the API Message Table:

Field	Type	Description
Business_Object_ID	VARCHAR2(3);	Error Handling API will be shared by ECO, BOM and RTG business objects. The default ID is ECO.
Message_Text	VARCHAR2(2000)	The actual message that the user sees. Please see below for format information.

Field	Type	Description
Entity_Id	VARCHAR2(3)	The entity that this message belongs to. This may hold BO, ECO, REV, RI, RC, RD, or SC. BO - Business Object ECO Header REV - ECO Revisions RI - Revised Items RC - Revised Components RD - Reference Designators SC - Substitute Components BH- Bills of Material Header BC - Bills of Material Comments
Entity_Index	NUMBER	The index of the entity array this record belongs to.

Message Formats

Expected errors and warnings: The message text contains the translated and token substituted message text. Please note that the message text may contain tokens, some of which will identify the entity instances that this message is for. The following tokens identify the several entities:

Assembly Item: Assembly_Item_Name

Revised Component: Revised_Component_Number

Substitute Component: Substitute_Component_Number

Reference Designator: Reference_Designator_Name

Assembly Comment: Standard_Remark_Designator

Unexpected errors:

<Package Name> <Procedure/Function Name> <SQL Error Number>

<SQL Error Message Text>

Other Message

An Other Message is a message that is logged for all records that are affected by an error in a particular record. So if an error in a assembly item record will cause all it's children to error out, then the following will be logged:

For the Assembly item itself, the error message describing the problem.

For all records affected by the type of error that occurred in the assembly item, the other message. This message essentially mentions the following:

How the error has affected this record, that is, it has been errored out too, with a severe or fatal error status, or that it has not been processed.

1. Which record caused this record to be affected.
2. What process flow step in the offending record caused this record to be affected.
3. What transaction type in the offending record caused this record to be affected.

Error Handler

The program performs all its error handling and messaging through the Error Handler. It makes calls to the Error Handler when an error or warning needs to be issued. The following are the functions of the Error Handler:

- Log the error/warning messages sent to it.
- Set the return status of the record in error.
- Set the return status of other records affected by this error.

The following is the input that the Error Handler expects from the calling program:

Input	Description
Business Object Identifier	Because the Error Handler will be shared by many business objects, the Business Object identifier will help identify the errors, especially when the same user executes the business object for BOM and ECO which share some of the entities.
Business Object Entity Records	Calling program must pass the whole business object as-is.

Input	Description
Message and Token List	List of messages generated for error in the current record. See below for description of this list.
Error Status	Status the record in error should be set to.
Error Level	Business Object hierarchy level that current record is an instance of. That is, the entity that the record in error belongs to.
Entity Array Index	Index of record in error in its encompassing entity array. Does not apply to ECO Header.
Error Scope	Indicates depth of error, that is, how many other records are affected by it.
Other Message and Token List	Message generated for the other affected records. See below for description.
Other Status	Status the other affected records should be set to.

The Message and Token List, and the Other Message and Token List are temporary arrays that the calling program maintains. They hold message-and-token records. The Error Handler must log these messages into the API Message List.

For expected errors and warnings, the translated message text is retrieved using the message name. Then, after any requested token translation is performed, the tokens are substituted into the translated message text, and the message is logged. For unexpected errors, the calling program itself sends a message text, so no message retrieval is needed. The message is logged after token translation and substitution is performed.

Field	Description
Message Name	Name of the message used to retrieve the translated message text. NULL for unexpected errors.
Message Text	Message text for unexpected errors.
Token Name	Name of the token in the message.

Field	Description
Token Value	Value of the token in the message.
Translate	Should this token value be translated ?

Since each message may have more than one token, the Message and Token List has as many occurrences of the same message as there are tokens in it. The same applies to the Other Message and Token List, except that this list needs to carry only one message which is assigned to all other affected records. Since this lone message may have more than one token, there may be more than one occurrence of the message in the list.

Please note that the API Message List is not public and must be accessed through the Messaging API's provided to access the message list.

These are the message list API's that can be used to perform various operations on the Message List:

Message API Name	Description
Initialize	This API will clear the message list.
Reset	This API will reset the message counter to the start of the message list.
Get_Message_List	This message list API will return a copy of the message list. It will be users responsibility to extract message from this copy.
Get_Entity_Message (IN p_entity_id, IN p_bo_identifier DEFAULT 'ECO' OUT x_message_list)	This API will return a list of messages for a requested entity and business object identifier.
Get_Entity_Message (IN p_entity_id, IN p_entity_index IN p_bo_identifier OUT x_message_text)	This API will return message from the index th position for an entity within a business object

Message API Name	Description
Delete_Message (IN p_entity_id, IN p_bo_identifier)	This API will delete all messages for a particular entity within a business object
Delete_Message (IN p_entity_ic, IN p_bo_identifier, IN p_entity_index)	This API will delete message from the index the position for an entity within a business object
Get_Message (OUT x_entity_id, OUT x_entity_index OUT x_message_text OUT x_bo_identifier	This API can be used within a loop to get one message at a time from the Message List. Every time the user does a get_message, a message counter will be incremented to the next message index.

Bill of Material Export API

The Bill of Material Export API provides the ability to export bill of material data for a particular assembly, in all subordinate organizations in a specified organization hierarchy. The number of levels to which a BOM is exploded for a particular organization depends on the Max Bill Levels field setting in the Organization Parameters form. If this value is greater than or equal to the levels of the bill being exported, then that bill will be exploded to the lowest level.

You can insert bill of material information returned by the API in the custom table or some other storage mechanism. This API supports companies having large organization structures.

Launching the Export

You need to call the API in the following way:

BOMPXINQ.EXPORT_BOM

INPUT Parameters

Profile_id

Security Profile Id.

Org_hierarchy_name

The name of the organization hierarchy to which all subordinate organizations will receive the exported bill of material data.

Assembly_item_id

Must be the inventory_item_id of the bill, and must exist in the mtl_system_items table for that organization. This item must exist in all subordinate organizations under the hierarchy origin.

Organization_id

Uniquely identifies a bill which will be exploded with the bill details in the bom_export_tab, PL/SQL table.

Alternate_bom_designator

The alternate bill defined for this primary bill. This can be passed as NULL or if there are no alternatives defined. It uniquely identifies a bill which will be exploded with the bill details in the bom_export_tab, PL/SQL table.

Costs

Pass parameter as 1, if cost details need to be exported. Pass the appropriate cost_type_id for that item and organization combination. If the parameter is passed as 2, then pass cost_type_id as having zero value. If this parameter is passed as NULL or, then it will take the default value of 2.

Cost_type_id

Pass the appropriate cost_type_id for that item and organization combination. This works in conjunction with the Costs parameter.

OUTPUT Parameters

bom_export_tab

PL/SQL table containing the exploded bill of material information. This information can be inserted into a custom table, written to a text file, or passed to host arrays (Oracle Call Interface.) Error_Code should have a value of zero and Err_Msg should be NULL, before inserting the date into a custom table.

Err_Msg

Error Messages.

Error_Code

Error Codes.

Export Error Handling and Messaging

The Bill of Material Export API may return the following values for error code:

Error Code	Description
0	Successful.
9998	Bill exceeds the maximum number of levels defined for that organization. You need to reduce the number of levels of the bill, or increase the maximum number of levels allowed for a bill in that organization.
SQLCODE	Oracle database related errors.

If the error code equals a value other than zero, the contents of the output PL/SQL table (bom_export_tab) are deleted. Because the output is inserted into a PL/SQL table instead of a database table, this API can be rerun multiple times without concerns regarding the committed data. After accessing this API, you should check the value of Error Code and Error Message before inserting the data from the PL/SQL table to custom tables.

PL/SQL Output Table (BOM_EXPORT_TAB) Columns

TOP_BILL_SEQUENCE_ID

BILL_SEQUENCE_ID

COMMON_BILL_SEQUENCE_ID

ORGANIZATION_ID

COMPONENT_SEQUENCE_ID

COMPONENT_ITEM_ID

COMPONENT_QUANTITY

PLAN_LEVEL

EXTENDED_QUANTITY

SORT_ORDER

GROUP_ID

TOP_ALTERNATE_DESIGNATOR

COMPONENT_YIELD_FACTOR

TOP_ITEM_ID

COMPONENT_CODE

INCLUDE_IN_ROLLUP_FLAG

LOOP_FLAG

PLANNING_FACTOR

OPERATION_SEQ_NUM

BOM_ITEM_TYPE

PARENT_BOM_ITEM_TYPE

ASSEMBLY_ITEM_ID

WIP_SUPPLY_TYPE

ITEM_NUM

EFFECTIVITY_DATE

DISABLE_DATE

IMPLEMENTATION_DATE

OPTIONAL

SUPPLY_SUBINVENTORY

SUPPLY_LOCATOR_ID

COMPONENT_REMARKS

CHANGE_NOTICE

OPERATION_LEAD_TIME_PERCENT

MUTUALLY_EXCLUSIVE OPTIONS

CHECK_ATP

REQUIRED_TO_SHIP

REQUIRED_FOR_REVENUE

INCLUDE_ON_SHIP_DOCS

LOW_QUANTITY

HIGH_QUANTITY

SO_BASIS

OPERATION_OFFSET

CURRENT_REVISION

LOCATOR

ATTRIBUTE1

ATTRIBUTE2

ATTRIBUTE3

ATTRIBUTE4

ATTRIBUTE5

ATTRIBUTE6

ATTRIBUTE7

ATTRIBUTE8

ATTRIBUTE9

ATTRIBUTE10

ATTRIBUTE11

ATTRIBUTE12

ATTRIBUTE13

ATTRIBUTE14

ATTRIBUTE15

Routing API Overview

The Routing API enables you to import routing information from a legacy system. You can create, update and delete routing information. You can automatically process routing data without inserting cryptic IDs or system specific information. It processes the information within a routing synchronously and stills you to process more than one routing simultaneously.

This section describes the basic business needs, major features, business object architecture and components for the routing API.

Routing API Features

Creating, Updating, and Deleting Routing Information

- Use the Routing API to create, update and delete a routing from a form and the external system. You can use the Routing API located in the Open Interface program to update or delete routing in the external system.

  When processing data, you do not have to provide system specific data. The Routing API requires only the business data needed to define the routing.

- Routing Data Encapsulation

  When processing data you do not have to provide system specific data. The Routing API requires only the business data needed to define the routing.

- Synchronous Processing of Information within a Routing.

  The Routing API processes the information within a routing synchronously. Following the business hierarchy, it processes operation sequences before operation resources.

- Asynchronous Processing of Routings.

  You can process different Routing business objects simultaneously. You do not have to wait for one process to finish before starting the next.

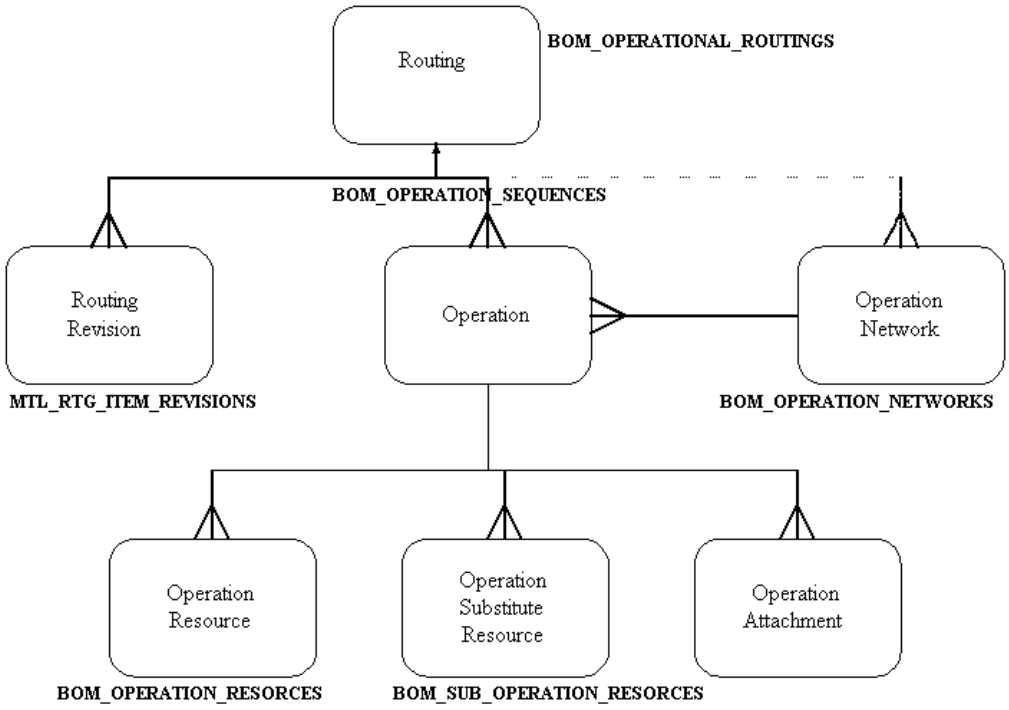
- Detailed and Translatable Error Messages.

  Upon failure, the Routing API reports detailed and translatable error messages. The error messages identifies the severity and scope of the error as well as how other associated records are effected.

Routing Entity Diagram

The following diagram shows the table structure to the Routing API along with the corresponding entities: (mfgap_3_6.gif)

Routing Entity Diagram



Routing Architecture

The Routing Business Object Architecture is based on the hierarchical definition of Routings and related sub-entities in Oracle Bill of Material. To use the Routing Business object interface, you only need to know the structure or your routing. As in a genealogical tree, the entity at the top is the parent. The entities connected directly below are its children.

Routing Header

The routing header entity defines the basic routing scheme. There can only be one routing header per routing scheme. The routing header contains information about the time, unit of measure, alternate designator, and the revision. To identify a routing, specify the item, the alternate designator, and the organization.

Routing Revisions

The revisions entity stores the routing revision information. To identify a routing revision specify the item and the organization.

Operation Sequence

The operation sequence entity enables you to specify an operation sequence. To identify an operation sequence, you must specify the item, alternate designator, organization, operation type, operation effectivity date, and operation sequence number.

Operation Resource

The operation resource entity enables you to add resources to an operation sequence. To identify an operation resource, specify the item, alternate designator, organization, operation type, operation sequence number, operation effectivity date, and resource sequence number.

Substitute Operation Resource

The substitute operation resource entity cannot exist without an operation sequence and related operation resources. Use this entity to add a substitute resource to t he operation sequence and operation resources that contain a substitute group number and scheduling sequence number. To identify a substitute operation resource, specify the item, alternate designator, organization operation sequence number, operation effectivity date, operation type, schedule sequence number, and substitute group number.

Operation Network

The operation network entity cannot exist without an operation sequence.

The Routing API as it exists in the database

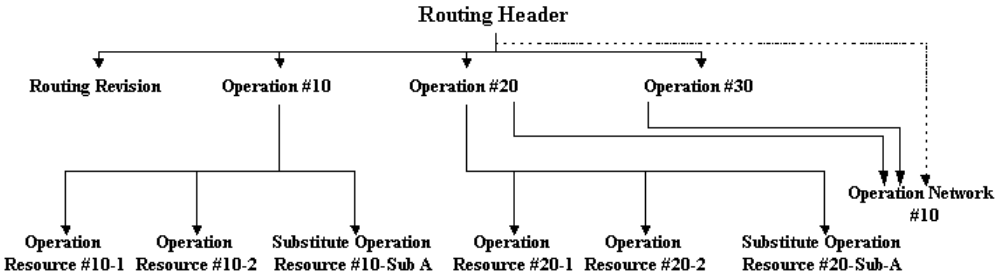
Each of the entities shown in the Routing entity diagram maps to a corresponding table in the database. The following are the existing production tables.

Production Table	Description
BOM_OPERATIONAL_ROUTINGS	Stores information about manufacturing and engineering routing headers
MTL_RTG_ITEM_REVISIONS	Stores information revisions levels for routings.
BOM_OEPRATION_RESOURCES	Stores information about resources that you require to complete operations on routings.
BOM_OPERATION_SEQUENCES	Stores information about operation sequences.
BOM_SUB_OPERATION_RESOURCES	Stores information about substitute resources used in an operation
BOM_OPERATION_NETWORKS	Stores information about routing operation network.

API Traversal Strategy

The routing API articular is a hierarchy of entities. You can also view it as a tree with branches, where the routing header entity is the parent, and each of the branches are child nodes or entities. Each node in a branch is the parent of the child nodes under it. Here is an example: (mfgapi_3_8.gif)

API Traversal Strategy



The Routing API is a closely tied object. If you do not have access to an assembly item, then you do not have access to any entity records associated with that item. Certain assembly item characteristics and errors affect the children and possibly other assembly items and other API records. To avoid these problems, the program must traverse the tree in and efficient manner to achieve the following:

- Cut down the amount of required processing. Use specific facts established during

process of other records. This saves the program from retrieving information again.

- Provide intelligent error handling. Sometimes an error in a parent record implies an error in all children attached to a parent. For example, if you are unable to create a routing header, you will not be able to create a revised operation for the routing.

Routing Import Description

Step	Purpose	Description	Error
Step 1: Pass Business Object to Public API	The program will try to import it	Caller must pass one business object at a time. There should be only one Header record. There may be more than one record for other entities.	-N/A-
Step 2: Check for Organization uniformity	We must ensure that all records in a business object belong to t he same Organization.	Set Business Object Identifier to 'RTG' in the system information. All records must have the same. Derive organization_id from Organization_Code Store organization_id value in System_Information record.	Severe Error I

Step	Purpose	Description	Error
Step 3: Save system information	Saves system-specific information in System_Information record since it is common to the whole business object. This information is stored in the database along with the record	Initialize User_Id, Login_Id, Prog_Appid, Prog_Id in System_Information record. Pull in values of profiles profile BOM: Standard Item Access, BOM: Model Item Access into STD_Item_Access, MDL_Item_Access and OC_Item_Access respectively. Pull in values of Cfm_routing_flag into Flow_Routing_Flag and Lot_Based_Routing_Flag.	Quit import of business object
Step 4: Pick up highest level un-processed record	The import program processes a parent and all its direct and indirect children, before moving onto to a sibling. So, the highest level parent must be chosen.	The highest level record with a {return status = NULL} is picked. When there are no more records, the program exits and transfers control to the caller.	-N/A-

Step	Purpose	Description	Error
Step 5: Convert user-unique index to unique index	Unique index helps uniquely identify a record in the database, and may consist of more than one column. User-unique index is a user-friendly equivalent of the unique index. It serves the following purposes:	The user need not enter cryptic Ids If a user unique index could not be derived for a parent, it's entire lineage is error-ed out since members of the lineage are referencing an invalid parent. Derive unique index columns from user-unique index columns.	Severe Error III
Step 6: Existence Verification	The record being updated or deleted in the database must already exist. But a record being created must not. Such an error in a record must cause all children to error out, since they are referencing an invalid parent	For CREATE, the record must not already exist. For UPDATE and DELETE, the record must exist. Query up database record into the associated OLD record.	Severe Error III

Step	Purpose	Description	Error
Step 7: Check Lineage	We must ensure that the linkage of records is correct in the business object. That is, child records must reference valid parents. A valid parent is one that exists, and is truly the current record's parent in the database.	Perform lineage checks for entity records that do not belong to the top-most entity in the hierarchy, based on Transaction_Type and the following factors: Immediate parent being referenced exists in the database, and, for UPDATE and DELETE, is truly the parent of this record in the database, OR if there is no immediate parent record in the business object, the indirect parent being referenced exists and is really the parent of the current record's parent in the database.	Severe Error III
Step 8(a): Check operability of parent items and current item (if applicable)	Routing and any of it's children (operation, resources etc.) cannot be operated upon if the assembly item is implemented or canceled.	Check if System_Information record has this information. If not, find it in the database assembly item record, and set System_Information flags accordingly.	Fatal Error III or Fatal Error II (depending on affected entity)

Step	Purpose	Description	Error
Step 8(b): Check assembly item operability for Routing	Routing and any of it's children (operation, resources etc.) cannot be operated upon if the user does not have access to assembly item for routing.	Check item attribute if the user can create routing for the assembly item. Check if System_Information record has this information. If not, find it in the database assembly item record, and set System_Information flags accordingly. Check if System_Information record has this information. If not, find it in the database assembly item record, and set System_Information flags accordingly.	Fatal Error II
Step 9: Check flow routing and Lot based routing operability	If the routing is flow routing (Cfm_routing_flag = 1). Check Flow Manufacturing have been installed If the routing is lot based routing (Cfm_routing_flag = 3). Check if the Organization is WSM Enabled	Check if System_Information record has this information. If not, find it in the database and set System_Information flag accordingly.	Fatal Error II

Step	Purpose	Description	Error
Step 10: Value-Id conversions	There are user-friendly value columns that derive certain Id columns. The value columns free up the user from having to enter cryptic Ids, since the Ids can be derived from them.	Derive Ids from user-friendly values.	CREATE: Severe Error IV. Other: Standard Error
Step 11: Required Fields checking	Some fields are required for an operation to be performed. Without them, the operation cannot go through. The user must enter values for these fields.	Check that the required field columns are not NULL.	CREATE: Severe Error IV. Other: Standard Error
Step 12: Attribute validation (CREATEs and UPDATEs)	Each of the attributes/fields must be checked individually for validity. Examples of these checks are: range checks and checks against lookups. .	Check that user-entered attributes are valid. Check each attribute independently of the others	CREATE: Severe Error IV. UPDATE: Standard Error
Step 13: Populate NULL column (UPDATEs and Deletes	The user may send in a record with certain values set to NULL. Values for all such columns are copied over from the OLD record. This feature enables the user to enter minimal information for the operation.	For all NULL columns found in business object record, copy over values from OLD record	-N/A-

Step	Purpose	Description	Error
Step 14: Default values for NULL attributes (CREATEs)	For CREATEs, there is no OLD record. So the program must default in individual attribute values, independently of each other. This feature enables you to enter minimal information for the operation to proceed.	For all NULL columns found in business object record, try to default in values, either by retrieving them from the database, or by having the program assign values.	Severe Error IV
Step 15: Check conditionally required attributes	Some attributes are required based on certain external factors such as the Transaction_Type value.	Perform checks to confirm all conditionally required attributes are present.	Severe Error IV
Step 16: Entity level defaulting	Certain column values may depend on profile options, other columns in the same table, columns in other tables, etc. Defaulting for these columns happens here.	For all NULL columns in record, try to default in values based on other values. Set all MISSING column values to NULL.	CREATE: Severe Error IV. UPDATE: Standard Error

Step	Purpose	Description	Error
Step 17: Entity level validation	This is where the whole record is checked. The following are checked: Non-updateable columns (UPDATEs): Certain columns must not be changed by the user when updating the record. Cross-attribute checking: The validity of attributes may be checked, based on factors external to it Business logic: The record must comply with business logic rules	Perform checks against record in the order specified in the -Purpose- column.	CREATE: Severe Error IV. UPDATE: Standard Error
Step 18: Database writes	Write record to database table.	Perform database write: Insert record for CREATE Overwrite record for UPDATE and CANCEl.	-N/A-
Step 19: Process direct and indirect children	The program will finish processing an entire branch before moving on to a sibling branch. A branch within the business object tree consists of all direct and indirect children.	Pick up the first un-processed child record and Go to Step 5. Continue until all direct children have been processed. Then pick up the first un-processed indirect child record and do the same as above. When no more records are found, Go to Step 20.	-N/A-

Step	Purpose	Description	Error
Step 20: Process siblings	When an entire branch of a record has been processed, the siblings of the current record are processed. The sibling may also contain a branch. So the processing for the sibling will be exactly the same as the current record.	Pick up the first un-processed sibling record and Go to Step 5. When no more records are found, Go to Step 21.	-N/A-
Step 21: Process parent record's siblings	Once all the siblings have been processed, the program moves up to the parent (of this entire branch) and processes all of its siblings (which contain branches of their own).	Go up to parents and pick up the first un-processed sibling of the parent. Go to Step 5. Continue through the loop until all siblings have been processed. When there are nor more records, Go to Step 4.	

Routing Parameter Descriptions

Routing Headers Exposed Columns

The following table describes all parameters the BOM_OPERATIONAL_ROUTINGS table uses. Additional information on these parameters follows.

IRouting Header Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	x	-	-
Organization_Code	VARCHAR2(3)	x	-	-

IRouting Header Name	Type	Required	Derived	Optional
Alternate_Routing_Code	VARCHAR2(10)	-	-	x
Eng_Routing_Flag	NUMBER	x	-	-
Common_Assembly_Item_Name	VARCHAR2(81)	-	X	-
Routing_Comment	VARCHAR2(240)	-	-	x
Completion_Subinventory	VARCHAR2(10)	-	-	x
Completion_Location_Name	VARCHAR2(81)	-	-	x
Line_Code	VARCHAR2(10)	-	-	x
CFM_Routing_Flag	NUMBER	-	-	x
Mixed_Model_Map_Flag	NUMBER	-	-	x
Total_Cycle_Time	NUMBER	-	-	x
Priority	NUMBER	-	-	x
CTP_Flag	NUMBER	-	-	x
Attribute_Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x

IRouting Header Name	Type	Required	Derived	Optional
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-

IRouting Header Name	Type	Required	Derived	Optional
Return_Status	VARCHAR2(1)	-	-	x
Original_System_Reference	VARCHAR2(50)	-	-	x
Delete_Group_Name	VARCHAR2(10)	-	-	x
DG_Description	VARCHAR2(240)	-	-	x

Assembly_Item_Name Inventory item name of a manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Organization_Code Identifies the organization

Default Value: FND_API.G_MISS_CHAR

Alternate_Routing_Code Alternate routing designator code

Default Value: FND_API.G_MISS_CHAR

Eng_Routing_Flag The rule that sets the routing as a manufacturing item or an engineering item. Valid values are as follows:

1. MFG Routing
2. Eng Routing

Default= FND_API.G_MISS_NUM

Common_Assembly_Item_Name Inventory item identifier of a common assembly. The default is derived from the common_assembly_item_name.

Default= FND_API.G_MISS_CHAR

Routing_Comment Common about Routing.

Default Value: FND_API.G_MISS_CHAR

Completion_Subinventory Destination subinventory for assembly

Default Value: FND_API.G_MISS_CHAR

Completion_Location_Name Destination location for an assembly

Default Value: FND_API.G_MISS_CHAR

Line_Code Name of the WIP line

Default Value: FND_API.G_MISS_CHAR

CFM_Routing_Flag Continuous flow, or traditional routing

Default Value: FND_API.G_MISS_NUM

Mixed_Model_Map_Flag You use this routing in mixed model map calculation. Valid values are as follows:

1. Yes
2. No)

Default Value: FND_API.G_MISS_NUM

Priority This parameter is for informational use.

Default Value: FND_API.G_MISS_NUM

Total_Cycle_Time Total time an assembly takes along the primary path in the operation network calculated by Flow Manufacturing.

Default Value= FND_API.G_MISS_NUM

CTP_Flag Flag that indicates capacity must be checked when item is ordered

Default Value: FND_API.G_MISS_NUM

Attribute_Category The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

Attribute 1-15 Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Delete_Group_Name Delete group name of the entity type you are deleting

Default Value: FND_API.G_MISS_CHAR

DG_Description Description of the group you are deleting.

Default Value: FND_API.G_MISS_CHAR

Routing Revision Exposed Columns

The following table describes all the parameters the MTL_RTG_ITEM_REVISIONS table.

IRouting Revision Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	x	-	-
Organization_Code	VARCHAR2(3)	x	-	-
Revision	VARCHAR2(3)	x	-	-
Start_Effective_Date	DATE	-	-	x
Attribute_Category	ARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	VARCHAR2(150	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150	-	-	x
Attribute8	VARCHAR2(150)	-	-	x

IRouting Revision Name	Type	Required	Derived	Optional
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150	-	-	x
Attribute14	ARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x
Transaction_Type-	VARCHAR2(30)	x	-	-
Return_Status-	VARCHAR2(1)	-	-	x
Original_System_Reference	VARCHAR2(50	-	-	x

Assembly_Item_Name Inventory item name of a manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Organization_Code Identifies the organization

Default Value: FND_API.G_MISS_CHAR

Alternate_Routing_Code Alternate routing designator code

Default Value: FND_API.G_MISS_CHAR

Revision Routing Revision

Default Value: FND_API.G_MISS_CHAR

Start_Effective_Date Effective Date

Default Value: FND_API.G_MISS_DATE

Attribute_category The category of the flexfield described in the attribute column

Default Value: FND_API.G_MISS_CHAR

Attribute 1-15 Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Operation Sequence Exposed Columns

The following table describes all the parameters the BOM_OPERATION_SEQUENCES table.

IOperation Sequence Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	x	-	-
Organization_Code	VARCHAR2(3)	x	-	-
Alternate_Routing_Code	VARCHAR2(10)	-	-	x
Operation_Sequence_Number	NUMBER	x	-	-
Operation_Type	NUMBER	x	-	-
Start_Effective_Date	DATE	-	-	x

IOperation Sequence Name	Type	Required	Derived	Optional
New_Operation_Sequence_Number	NUMBER	-	-	x
New_Start_Effective_Date	DATE	-	-	x
Standard_Operation_Code	NUMBER	x	-	-
Department_Code	NUMBER	-	x	-
Op_Lead_Time_Percent	NUMBER	-	-	x
Minimum_Transfer_Quantity	NUMBER	-	-	x
Count_Point_Type	NUMBER	-	-	x
Operation_Description	VARCHAR2(240)	-	-	x
Disable_Date	DATE	-	-	x
Backflush_Flag	NUMBER	-	-	x
Option_Dependent_Flag	NUMBER	-	-	x
Reference_Flag	NUMBER	-	-	x
Process_Seq_Number	NUMBER	-	-	x
Process_Code	VARCHAR2(4)	-	-	x
Line_Op_Code	VARCHAR2(4)	-	-	x

IOperation Sequence Name	Type	Required	Derived	Optional
Yield	NUMBER	-	-	x
Cumulative_Yield	NUMBER	-	-	x
Reverse_CUM_Yield	NUMBER	-	-	x
User_Labor_Time	NUMBER	-	-	x
User_Machine_Time	NUMBER	-	-	x
Net_Planning_Percent	NUMBER	-	x	-
Include_In_Rollup	NUMBER	-	-	x
Op_Yield_Enabled_Flag	NU,MBER	-	-	x
Attribute_Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x

IOperation Sequence Name	Type	Required	Derived	Optional
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x
Transaction_Type	VARCHAR2(30))	x	-	-
Return_Status	VARCHAR2(1)	-	-	x

Assembly_Item_Name Inventory item name of a manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Organization_Code Identifies the organization.

Default Value: FND_API.G_MISS_CHAR

Alternate_Routing_Code Alternate routing designator code.

Default Value: FND_API.G_MISS_CHAR

Operation_Sequence_Number Routing operation unique identifier

Default Value: FND_API.G_MISS_NUM

Operation_Type A process, a line operation, or an event.

Default Value: FND_API.G_MISS_NUM

Start_Effective_Date

Default Value: FND_API.G_MISS_DATE

New_Operation_Sequence_Number Added to facilitate updates because it is part of the primary key.

Default Value: FND_API.G_MISS_NUM

New_Start_Effective_Date Added to facilitate updates because it is part of the primary key.

FND_API.G_MISS_DATE

Standard_Operation_Code Standard operation unique identifier

Default Value: FND_API.G_MISS_CHAR

Department_Code Department Code.

Default Value: FND_API.G_MISS_CHAR

Op_Lead_Time_Percent Indicates the amount of overlapping lead time the child operation with the parent operation.

Default Value: FND_API.G_MISS_NUM

Minimum_Transfer_Quantity Minimum operation transfer quantity.

Default Value: FND_API.G_MISS_NUM

Count_Point_Type Operation Move Type

FND_API.G_MISS_NUM

Operation_Description Description of the operation

FND_API.G_MISS_CHAR

Disable_Date Date the operation is no longer effective. Effectivity lasts until the end of the disable date.

Default Value: FND_API.G_MISS_DATE

Backflush_Flag Indicates if an operations requires Backflush_Flag.

Default Value: FND_API.G_MISS_NUM

Option_Dependent_Flag Indicates if to sue this operation in all configuration routings, even if no components of the configuration are used in this operation.

Default Value: FND_API.G_MISS_NUM

Reference_Flag If the Standard operation is references or copied then the operation cannot be updated.

Default Value: FND_API.G_MISS_NUM

Process_Seq_Number Operation sequence identifier of parent process. This applies only to events.

Default Value: FND_API.G_MISS_NUM

Process_Code Process code.

Default Value: FND_API.G_MISS_CHAR

Line_Op_Seq_Number Operation sequence identifier of the parent line operation. This applies only to events.

Default Value: FND_API.G_MISS_NUM

Line_Op_Code Line Operation code.

FND_API.G_MISS_CHAR

Yield Process Yield at this operation.

Default Value: FND_API.G_MISS_NUM

Cumulative_Yield Cumulative process yield from beginning of routing to this operation.

Default Value: FND_API.G_MISS_NUM

Reverse_CUM_Yield Cumulative process yield from end of routing to comparable operation.

Default Value: FND_API.G_MISS_NUM

User_Labor_Time User calculated run time attributable to labor.

Default Value: FND_API.G_MISS_NUM

User_Machine_Time User calculated run time attributable to machines.

Default Value: FND_API.G_MISS_NUM

Net_Planning_Percent Cumulative planning percent derived from the operation network.

Default Value: FND_API.G_MISS_NUM

Include_In_Rollup Indicates if the operation yield is considered in cost rollup.

Default Value: FND_API.G_MISS_NUM

Op_Yield_Enabled_Flag Indicates if operation yield is considered during costing.

Default Value: FND_API.G_MISS_NUM

Attribute_category The category of the flexfield described in the attribute column

Default Value: FND_API.G_MISS_CHAR

Attribute 1-15 Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Delete_Group_Name Delete group name of the entity type you are deleting.

Default Value: FND_API.G_MISS_CHAR

DG_Description Description of the group you are deleting.

Default Value: FND_API.G_MISS_CHA

Operation Resources Exposed Columns

The following table describes all the parameters the BOM_OPERATION_RESOURCES table.

Operation Resource Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	x	-	-
Organization_Code	VARCHAR2(3)	x	-	-
Alternate_Routing_Code	VARCHAR2(10)	-	-	
Operation_Sequence_Number	NUMBER	-	-	x
Operation_Type	NUMBER	x	-	-

Operation Resource Name	Type	Required	Derived	Optional
Op_Start_Effective_Date	DATE	x	-	-
Resource_Sequence_Number	NUMBER	x	-	-
Resource_Code	VARCHAR2(10)	-	-	x
Activity	VARCHAR2(10)	-	x	-
Standard_Rate_Flag	NUMBER	-	x	-
Assigned_Units	NUMBER	-	x	-
Usage_Rate_Or_amount	NUMBER	-	-	x
Usage_Rate_Or_Amount_Inverse	NUMBER	-	-	x
Basis_Type	NUMBER	-	x	-
Schedule_Flag	NUMBER	-	x	-
Resource_Offset_Percent	NUMBER	-	-	x
Autocharge_Type	NUMBER	-	x	-
Schedule_Sequence_Number	NUMBER	-	-	x
Principle_Flag	NUMBER	-	-	x
Setup_Type	VARCHAR2(30)	-	-	x
Attribute_Category	VARCHAR2(30)	-	-	x

Operation Resource Name	Type	Required	Derived	Optional
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute7	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x

Operation Resource Name	Type	Required	Derived	Optional
Attribute15	VARCHAR2(150)	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Return_Status	VARCHAR2(1)	-	-	x
Original_System_Reference	VARCHAR2(50)	-	-	x

Assembly_Item_Name Inventory item name of a manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Organization_Code Identifies the organization.

Default Value: FND_API.G_MISS_CHAR

Alternate_Routing_Code Itinerant routing designator code.

Default Value: FND_API.G_MISS_CHAR

Operation_Sequence_Number Routing operation unique identifier

Default Value: FND_API.G_MISS_NUM

Operation_Type A process, a line operation, or an event.

Default Value: FND_API.G_MISS_NUM

Start_Effective_Date

Default Value: FND_API.G_MISS_DATE

New_Operation_Sequence_Number Added to facilitate updates because it is part of the primary key.

Default Value: FND_API.G_MISS_NUM

New_Start_Effective_Date Added to facilitate updates because it is part of the primary key.

FND_API.G_MISS_DATE

Standard_Operation_Code Standard operation unique identifier.

Default Value: FND_API.G_MISS_CHAR

Department_Code Department code.

Default Value: FND_API.G_MISS_CHAR

Op_Lead_Time_Percent Indicates the amount of overlapping lead time the child operation with the parent operation.

Default Value: FND_API.G_MISS_NUM

Minimum_Transfer_Quantity Minimum operation transfer quantity.

Default Value: FND_API.G_MISS_NUM

Count_Point_Type Operation Move Type.

FND_API.G_MISS_NUM

Operation_Description Description of the operation.

FND_API.G_MISS_CHAR

Disable_Date Date the operation is no longer effective. Effectivity lasts until the end of the disable date.

Default Value: FND_API.G_MISS_DATE

Backflush_Flag Indicates if an operations requires Backflush_Flag.

Default Value: FND_API.G_MISS_NUM

Option_Dependent_Flag Indicates if to sue this operation in all configuration routings, even if no components of the configuration are used in this operation.

Default Value: FND_API.G_MISS_NUM

Reference_Flag If the Standard operation is references or copied then the operation cannot be updated.

Default Value: FND_API.G_MISS_NUM

Process_Seq_Number Operation sequence identifier of parent process. This applies only to events.

Default Value: FND_API.G_MISS_NUM

Process_Code Process code.

Default Value: FND_API.G_MISS_CHAR

Line_Op_Seq_Number Operation sequence identifier of the parent line operation. This applies only to events.

Default Value: FND_API.G_MISS_NUM

Line_Op_Code Line operation code.

FND_API.G_MISS_CHAR

Yield Process yield at this operation.

Default Value: FND_API.G_MISS_NUM

Cumulative_Yield Cumulative process yield from beginning of routing to this operation.

Default Value: FND_API.G_MISS_NUM

Reverse_CUM_Yield Cumulative processes yield from end of routing to comparable operation.

Default Value: FND_API.G_MISS_NUM

User_Labor_Time User calculated run time attributable to labor.

Default Value: FND_API.G_MISS_NUM

User_Machine_Time User calculated run time attributable to machines.

Default Value: FND_API.G_MISS_NUM

Net_Planning_Percent Cumulative planning percent derived from the operation network.

Default Value: FND_API.G_MISS_NUM

Include_In_Rollup Indicates if the operation yield is considered in cost rollup.

Default Value: FND_API.G_MISS_NUM

Op_Yield_Enabled_Flag Indicates if operation yield is considered during costing.

Default Value: FND_API.G_MISS_NUM

Attribute_category The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

Attribute 1-15 Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

Original_System_Reference Legacy system from which the data for the current record has come.

Default Value: FND_API.G_MISS_CHAR

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Delete_Group_Name Delete group name of the entity type you are deleting.

Default Value: FND_API.G_MISS_CHAR

DG_Description Description of the group you are deleting.

Default Value: FND_API.G_MISS_CHA INDEX BY BINARY_INTEGER

Substitute Operation Resources Exposed Columns

The following table describes all the parameters the BOM_SUB_OPERATION_RESOURCES table.

Substitute Operation Resource Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	x	-	-
Organization_Code	VARCHAR2(3)	x	-	-
Alternate_Routing_Code	VARCHAR2(10)	-	-	x
Operation_Sequence_Number	NUMBER	x	-	-
Operation_Type	NUMBER	x	-	-
Op_Start_Effective_Date	DATE	x	-	-
Sub_Resource_Code	VARCHAR2(10)	x	-	-
New_Sub_Resource_Code	VARCHAR2(10)	-	-	x
Schedule_Sequence_Number	NUMBER	x	-	-
Replacement_Group_Number	NUMBER	x	-	-
Activity	VARCHAR2(10)	-	x	-
Standard_Rate_Flag	NUMBER	-	x	-
Assigned_Units	NUMBER	-	x	-

Substitute Operation Resource Name	Type	Required	Derived	Optional
Usage_Rate_Or_amount	NUMBER	-	-	x
Usage_Rate_Or_Amount_Inverse	NUMBER	-	-	x
Basis_Type	NUMBER	-	x	-
Schedule_Flag	NUMBER	-	x	-
Resource_Offset_Percent	NUMBER	-	-	x
Autocharge_Type	NUMBER	-	x	-
Principle_Flag	NUMBER	-	-	x
Setup_Type	VARCHAR2(30)	-	-	x
Attribute_Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x

Substitute Operation Resource Name	Type	Required	Derived	Optional
Attribute7	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x
Transaction_Type	VARCHAR2(30)	-	-	x
Return_Status	VARCHAR2(1)	-	-	x

Assembly_Item_Name Inventory item name of a manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Organization_Code Identifies the organization.

Default Value: FND_API.G_MISS_CHAR

Alternate_Routing_Code Itinerant routing designator code.

Default Value: FND_API.G_MISS_CHAR

Operation_Sequence_Number Routing operation unique identifier

Default Value: FND_API.G_MISS_NUM

Operation_Type A process, a line operation, or an event.

Default Value: FND_API.G_MISS_NUM

Op_Start_Effective_Date Start date of the operation.

Default Value:FND_API.G_MISS_DATE

Sub_Resource_Code Substitute resource identifier.

Default Value: FND_API.G_MISS_CHAR

New_Sub_Resource_Code

Default Value: FND_API.G_MISS_CHAR

Schedule_Sequence_Number Scheduling Sequence Number

Default Value: FND_API.G_MISS_NUM

Replacement_Group_Number Substitute Group Number

Default Value: FND_API.G_MISS_NUM

Activity Activity identifier

Default Value FND_API.G_MISS_CHAR

Standard_Rate_Flag Use standard rate for shopfloor transactions

Default Value FND_API.G_MISS_NUM

Assigned_Units Resource units assigned

Default Value: FND_API.G_MISS_NUM

Usage_Rate_Or_Amount Resource usage rate.

Default Value: FND_API.G_MISS_NUM

Usage_Rate_Or_Amount_Inverse Resource usage rate inverse.

Default Value: FND_API.G_MISS_NUM

Basis_Type Basis type identifier.

Default Value: FND_API.G_MISS_NUM

Schedule_Flag

Default Value: FND_API.G_MISS_NUM

Resource_Offset_Percent Resource offset percent from the start of the routing.

Default Value: FND_API.G_MISS_NUM

Autocharge_Type

Default Value: FND_API.G_MISS_NUM

Principle_Flag Principle Flag

Default Value: FND_API.G_MISS_NUM

Attribute_Category The category of the flexfield described in the attribute column.

Default Value: FND_API.G_MISS_CHAR

Attribute 1-15 Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

Setup_Type Setup ID

Default Value:

Original_System_Reference Legacy system from which the data from the current record has come.

Transaction_Type Type of action including create, update or delete.

Default Value: FND_API.G_MISS_CHAR

Return_Status Processing status of the API after it completes its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API.G_RET_STS_ERROR

Operation Networks Exposed Columns

The following table describes all the parameters the BOM_OPERATION_NETWORKS table.

Operation Networks Name	Type	Required	Derived	Optional
Assembly_Item_Name	VARCHAR2(81)	x	-	-
Organization_Code	VARCHAR2(3)	x	-	-
Alternate_Routing_Code	VARCHAR2(10)	-	-	x
Operation_Type	NUMBER	-	-	x
From_Op_Seq_Number	NUMBER	x	-	-

Operation Networks Name	Type	Required	Derived	Optional
From_Start_Effective_Date	DATE	-	-	x
To_Op_Seq_Number	NUMBER	x	-	-
To_Start_Effective_Date	DATE	-	-	x
Connection_Type	NUMBER	-	-	x
Planning_Percent	NUMBER	-	-	x
Attribute_Category	VARCHAR2(30)	-	-	x
Attribute1	VARCHAR2(150)	-	-	x
Attribute2	VARCHAR2(150)	-	-	x
Attribute3	VARCHAR2(150)	-	-	x
Attribute4	VARCHAR2(150)	-	-	x
Attribute5	VARCHAR2(150)	-	-	x
Attribute6	VARCHAR2(150)	-	-	x
Attribute8	VARCHAR2(150)	-	-	x
Attribute9	VARCHAR2(150)	-	-	x

Operation Networks Name	Type	Required	Derived	Optional
Attribute10	VARCHAR2(150)	-	-	x
Attribute11	VARCHAR2(150)	-	-	x
Attribute12	VARCHAR2(150)	-	-	x
Attribute13	VARCHAR2(150)	-	-	x
Attribute14	VARCHAR2(150)	-	-	x
Attribute15	VARCHAR2(150)	-	-	x
Transaction_Type	VARCHAR2(30)	x	-	-
Return_Status	VARCHAR2(1)	-	-	x
Original_System_Reference	VARCHAR2(50)	-	-	x

Assembly_Item_Name Inventory item name of a manufactured assembly.

Default Value: FND_API.G_MISS_CHAR

Organization_Code Identifies the organization.

Default Value: FND_API.G_MISS_CHAR

Alternate_Routing_Code Alternate routing designator code.

Default Value: FND_API.G_MISS_CHAR

Operation_Type A process, a line operation, or an event

Default Value: FND_API.G_MISS_NUM

From_Op_Seq_Number From routing operation identifier

FND_API.G_MISS_NUM

From_Start_Effective_Date TBD

Default Value: FND_API.G_MISS_DATE

To_Op_Seq_Number To routing operation identifier.

Default Value: FND_API.G_MISS_NUM

To_Start_Effective_Date TBD

Default Value: FND_API.G_MISS_DATE

Connection_Type Primary, Alternate or Rework connection

Default Value: FND_API.G_MISS_NUM

Planning_Percent Planning Percentage

Default Value: FND_API.G_MISS_NUM

Attribute_Category The category of the flexfield described in the attribute column

Default Value: FND_API.G_MISS_CHAR

Attribute 1-15 Descriptive text for flexfields.

Default Value: FND_API.G_MISS_CHAR

Return Status Processing status of the API after it complete its function. Valid Values include:

Success: FND_API.G_MISS_CHAR

Error: FND_API_G_RET_STS_ERROR

Original_System_Reference Legacy system from which the data from the current record has come.

Transaction_Type Type of action including create, update or delete.

Launching the Routing Import

Three Step Rule

You must follow the three step rule to use the Routing API effectively. The following section describes the steps in detail.

1. Initialize the system information.
2. Call the Public API.
3. Review all relevant information after the Public API completes.

Step 1: Initialize the System Information

Each database table the programs writes to requires system information. You initialize specific variables to provide this information to the import program. The program

retrieves system information from these variables during operation. Use the following procedure to initialize the variables:

FND_BLGOAL.apps_initialize

(user_IDIN NUMBER

,resp_idIN NUMBER

,resp_appl_idIN NUMBER]

,security_group_idINNUMBER)

Pointers

1. This procedure initializes the global security context for each database session.
2. This initialization should be done when the session is established outside of a normal forms or concurrent program connection.
3. User_id is the FND User ID of the person launching the program.
4. Resp_id is the FND Responsibility Id.
5. Resp_appl_id is the application Responsibility ID.
6. Security_group is the FND Security Group ID.

Step 2: Call the Public API

The Public API is your interface to the Import program. You must call the API programmatically, while sending in one business object at a time. The Public API returns the process business object, the business object status, and a count of all associated err and warning messages. The procedure to call the API is as follows:

PROCEDURE Process_Rtg

(p_bo_identifier IN VARCHAR2:= 'RTG'

, p_api_version_number IN NUMBER:= 1.0

, p_init_msg_list IN BOOLEAN:= FALSE

, p_rtg_header_rec IN Bom_Rtg_Pub.Rtg_Header_Rec_Type

:=Bom_Rtg_Pub.G_MISS_RTG_HEADER_REC

, p_rtg_revision_tbl IN Bom_Rtg_Pub.Rtg_Revision_Tbl_Type

:=Bom_Rtg_Pub.G_MISS_RTG_REVISION_TBL

, p_operation_tbl IN Bom_Rtg_Pub.Operation_Tbl_Type

:= Bom_Rtg_Pub.G_MISS_OPERATION_TBL

, p_op_resource_tbl IN Bom_Rtg_Pub.Op_Resource_Tbl_Type

:= Bom_Rtg_Pub.G_MISS_OP_RESOURCE_TBL

, p_sub_resource_tbl IN Bom_Rtg_Pub.Sub_Resource_Tbl_Type

:= Bom_Rtg_Pub.G_MISS_SUB_RESOURCE_TBL

, p_op_network_tbl IN Bom_Rtg_Pub.Op_Network_Tbl_Type

:= Bom_Rtg_Pub.G_MISS_OP_NETWORK_TBL

, x_rtg_header_rec OUT Bom_Rtg_Pub.Rtg_Header_Rec_Type

, x_rtg_revision_tbl OUT Bom_Rtg_Pub.Rtg_Revision_Tbl_Type

, x_operation_tbl OUT Bom_Rtg_Pub.Operation_Tbl_Type

, x_op_resource_tbl OUT Bom_Rtg_Pub.Op_Resource_Tbl_Type

, x_sub_resource_tbl OUT Bom_Rtg_Pub.Sub_Resource_Tbl_Type

, x_op_network_tbl OUT Bom_Rtg_Pub.Op_Network_Tbl_Type

, x_return_status OUT VARCHAR2

, x_msg_count OUT NUMBER

, p_debug IN VARCHAR2:= 'N'

, p_output_dir IN VARCHAR2:= NULL

, p_debug_filename IN VARCHAR2:= 'RTG_BO_debug.log'

As is obvious from the above specification, all IN parameters begin with *p_*. All OUT parameters begin with *x_*. The following is a description of these parameters:

- p_api_version_number: It is used by the API to compare the version numbers of incoming calls to its current version number, and return an unexpected error if they are incompatible. See section 4.1 of the Business Object Coding Standards document for a detailed description of this parameter.

- p_init_msg_list: This parameter, if set to TRUE, allows callers to request that the API do the initialization of the message list on their behalf. On the other hand, the caller may set this to FALSE (or accept the default value) in order to do the initialization itself by calling FND_MSG_PUB.Initialize.

- p_rtg_header_rec, p_rtg_revision_tbl, p_operation_tbl, p_op_resource_tbl, p_sub_resource_tbl, p_op_network_tbl: This is the set of data structures that represents the incoming business object. p_rtg_header_rec is a record that holds the Routing header for the Routing. All the other data structures are PL/SQL tables of records that hold records for each of the other entities. All these data structures directly correspond to the entities shown in the Routing entity diagram.

Please note that any of these data structures may be sent in empty (set to NULL) to indicate that there are no instances of that entity in the business object being sent in.

- x_rtg_header_rec, x_rtg_revision_tbl, x_operation_tbl, x_op_resource_tbl,

x_sub_resource_tbl, x_op_network_tbl: This is the set of data structures that represents the outgoing business object. These records essentially constitute the whole business object as it was sent in, except now it has all the changes that the import program made to it through all the steps in the Process Flow. These records can be committed, rolled back, or subjected to further processing by the caller. x_rtg_header_rec is a record that holds the Routing header for the Routing. All the other data structures are PL/SQL tables of records that hold records for each of the other entities. All these data structures directly correspond to the entities shown in the Routing entity diagram.

- x_return_status: This is a flag that indicates the state of the whole business object after the import. If there has been an error in a record, this status will indicate the fact that there has been an error in the business object. Similarly, if the business object import has been successful, this flag will carry a status to indicate that. The caller may look up this flag to choose an appropriate course of action (commit, rollback, or further processing by the caller).

- x_msg_count: This holds the number of messages in the API message stack after the import. This parameter returns a 0 when there are no messages to return.

- x_msg_data: This returns a single message when the message count is 1. The purpose of this parameter is to save the caller the extra effort of retrieving a lone message from the message list. This parameter returns NULL when there is more than one message.

As mentioned above, the Public API must be called programmatically. The caller here may be a form, a shell, or some other API which serves to extend the functionality of the import program. Please see the Sample Shell section in the Appendix for a complete listing of the shell that was written to test the import program. This shell illustrates the correct usage of the import program.

> **Note:** A record must have an error status of NULL for it to be processed. If it has any other value, it will not be picked up for processing. The user must remember to NULL out this field when sending in a record.

Step 3: Review all relevant information after the Public API has completed

You must look up the following

- all error status that the Public API returns, including, the overall business object error status, as well as the individual record statuses.

- all record attributes to see what changes occurred after applying business logic to these records.

- all error and warning messages in the API message list.

The user can access the API message list using the following procedures and functions in the Error_Handler package:

Resetting Messages:

The following commands allow you to reset messages

- PROCEDURE Initialize
- PROCEDURE Reset;

Retrieving Messages:

The following commands enable you to retrieve messages

- PROCEDURE Get_Message_List
- (x_message_list OUT Error_Handler.Error_Tbl_Type);
- PROCEDURE Get_Entity_Message
- (p_entity_id IN VARCHAR2
- , x_message_list OUT Error_Handler.Error_Tbl_Type);
- PROCEDURE Get_Entity_Message
- (p_entity_id IN VARCHAR2
- p_entity_index IN NUMBER ,
- x_message_text OUT VARCHAR2);
- PROCEDURE Get_Message (x_message_text OUT VARCHAR2 ,
- x_entity_index OUT NUMBER
- , x_entity_id OUT VARCHAR2);
- FUNCTION Get_Message_Count RETURN NUMBER;

Deleting Messages:

The following commands enable you to delete messages:

- PROCEDURE Delete_Message

- (p_entity_id IN VARCHAR2 , p_entity_index IN NUMBER);
- PROCEDURE Delete_Message
- (p_entity_id IN VARCHAR2);
- PROCEDURE Dump_Message_List;

Routing Package Interaction

The Public Package - BOM_Rtg_PUB

This package is like a gatekeeper, letting only one business object through at a time. This essentially means that all records in the business object must *belong to* the business object. The business object here is the Routing, and incoming records together make up an instance of the business object. So, all records in an Routing Business Object instance must reference the same Routing.

Main Procedure: Process_Rtg

Set business object identifier to RTG in the System Information record.

1. Set business object status to S.
2. Check that all records in the business object belong to the same Routing, i.e., all records must have the same Assembly Item Name and Organization_Code combination and Alternate Designator.

Description	Cause of Failure	Error	Message
If there is an Routing Header in the business object, check that all records have the same Assembly_Item_Name, Organization_Code and Alternate Designator values as the Header. If the business object does not have an Routing Header record, check that all records have the same Assembly_Item_Name and Organization_Code combination as the first highest level entity record picked up.	Any records have mismatched Assembly_Item_Name, Organization_Code and Alternate_Routing_Code values	Severe Error I	BOM_MUST_BE_IN_SAME_RTG All records in a business object must belong to the same Routing. That is, they must all have the same Assembly Item Name, Organization and Alternate Designator. Please check your records for this.

3. Derive Organization_Id from Organization_Code and copy this value into all business object records.

Column	Description	Error	Message
Organization_Id	Derive using Organization_Code from table MTL_PARAMTERS	Severe Error I	BOM_ORG_INVALID: The Organization <rigid> you entered is invalid.

Unexpected Error Other Messages

BOM_UNEXP_ORG_INVALID

Due to an unexpected error, while performing a value to id conversion for the organization code, this record was not processed.

Pass business object into Private API if the business object status is still S. Also pass the Assembly Item Name and Organization_Id to Private API, to identify the business object instance.Accept processed business object and return status from Private API after the import, and pass it back to the calling program.

Routing Import Error Handling and Messaging

Error Handling Concepts

Error handling depends on the severity of the error, the entities the error extends itself over (*scope* of the error), and how the error affects the lineage (child record error states). When an error occurs, records are marked so that erroneous records are not processed again.

Error Severity Levels Severity levels help distinguish between different types of errors since the import program behaves differently for each of these errors. The following is a list of the error severity levels recognized by the import program:

Error Severity

CODE	MEANING
'W'	Warning / Debug
'E'	Standard Error
'E'	Severe Error
'F'	Fatal Error
'U'	Unexpected error

Error States serves two purposes:

- They convey to the user the exact type of error in the record.
- They help the import program identify the records that do not need to be processed.

Error States

CODE	MEANING
'S'	Success
'E'	Error
'F'	Fatal Error
'U'	Unexpected Error
'N'	Not Processed

Error Scope:

This indicates what the depth of the error is in the business object, that is, how many other records in the business object hierarchy the current error affects.

Error Scope

CODE	MEANING
'R'	Error affects current 'Record
'S'	Error affects all 'Sibling and child records
'C'	Error affects 'Child records
'A'	Error affects 'All records in business object

Child Error States:

If an error in a record affects child records, the status of the child may vary based on the type of error. There are two error states that indicate how the child is affected:

Child Error States

CODE	MEANING
'E'	Error

CODE	MEANING
'N'	Not Processed

Error Classes

There are three major classes that determine the severity of the problem.

Expected errors: These are errors the program specifically looks for in the business object, before committing it to the production tables.

1. Standard Error: This error causes only the current record to be error-ed out, but is not serious enough to affect any other records in the object. The current record status is set to E.

2. Severe Error I: This error affects all records. All record statuses are set to E. This error is usually a change notice/organization uniformity error. All records must have the same change notice/organization combination.

3. Severe Error II: This error affects all records that are children of this record's parent, when the parent is not in the business object. A characteristic of this record's parent caused the error, so all it's siblings and their children also get a record status of E. This error usually occurs when a lineage check fails.

4. Severe Error III: This error not only affects the current record but also its child records in the business object. The child record statuses are set to E. Please check your child records for errors as well as the current record. This error is usually a user-unique to unique index conversion error.

5. Severe Error IV: This error affects the current record and its child records since the program cannot properly process the child records. The child record statuses are set to N. This type of errors occur when there are errors on CREATEs

6. Fatal Error I: These errors occur when it is not possible to perform any operation on the Routing. Such errors affect the entire business object. All record statuses are set to F. The following are situations that cause this error:

   - You do not have access to this Item Type: (BOM is not allowed, PTO Item)

   - You do not have access to Flow Routing or Lot Based Routing.

7. Fatal Error II: This error affects all records that are children of this record's parent, when the parent is not in the business object. A characteristic of this record's parent caused the error, so all it's siblings and their children also get a record status of F.

8. Fatal Error III: These errors affects the current record and its children, since it is not possible to perform any operation on these records. The current record and all its child record statuses are set to F. The following situations cause this error:

    - You do not have access to this assembly item BOM item type.

Unexpected errors: All errors that are not expected errors are unexpected errors. These are errors that the program is not specifically looking for, for example, the user somehow loses the database connection.

Warnings: These are messages for information only. The purpose of warnings is:

to warn the user of problems that may occur when the routing is implemented. For example: the user entered value in the column only for Flow Routing is ignored in a standard routing.

to inform the user of any side-effects caused by user-entered data. For example: New Delete Group is created.

In order to bring together all the concepts above into a workable algorithm, we must introduce some terms that used extensively in this section, and the rest of the document.

Child record: All records carry the unique keys for all parents above them. A child record (of a particular parent) is one that holds the same values for the unique key columns as the parent record.

Sibling record: A sibling record (of the current record) is one that holds the same parent unique key column values as the current record. For example, a operation record that holds the same parent routing unique key column values as the current operation record, is a sibling of the current operation record. Likewise, an operation resource record that holds the same parent operation sequence unique key column values is a sibling of the operation resource

Business Object Status: This indicates the state of the whole business object after the import. As soon as the import program encounters an erroneous record, it sets the business object error status (also called return status) appropriately to convey this to the user. It is then up to the user to locate the offending record(s) using the individual record error statuses as indicated below. The caller may also use the business object return status to choose an appropriate course of action (commit, rollback, or further processing by the caller).

The following is a list of all the possible business object states:

Error Status

CODE	MEANING
'S'	Success

CODE	MEANING
'E'	Error
'F'	Fatal Error
'U'	Unexpected Error

Record Error Status: This is the state of the record after the import (success or error). The error status is also referred to as *return status* or *error state* in this document. Please see the Error States section above for a list of statuses a record can receive. The error status helps locate erroneous records in a business object that has error-ed out. The following are important pointers about the record error status.

- Every record is assigned an error status by the import program. Hence, if a record has a NULL return status, it means that the import program has not gotten to it yet.
- The user must send in records with {return status = NULL}. The import program will not look at records that already have an error status value, since it assumes that a record with an error status value has already been looked at by the program.

The following shows how the error status, error scope, and child statuses relate together to constitute the different error classes for records:

Error	Status	Scope	Child Statuses
Warning	S: Success	R: Record Only	-N/A-
Standard Error	E: Error	R: Record Only	-N/A-
Severe Error I	E: Error	A: All Records	E: Error
Severe Error II	E: Error	S: Current, Sibling and Child Records	E: Error
Severe Error III	E: Error	C: Current and Child Record	E: Error
Severe Error IV	E: Error	C: Current and Child Records	N: Not Processed
Fatal Error I	F: Fatal Error	A: All Records	

Error	Status	Scope	Child Statuses
Fatal Error II	F: Fatal Error	S: Current, Sibling and Child Records	
Fatal Error III	F: Fatal Error	C: Current and Child Record	
Unexpected Error	U: Unexpected Error	-N/A-	N: Not Processed

API Messaging

API Message Table

All messages are logged in the API Error Message Table. This is a PL/SQL table (array) of messages. Please see *Accessing Messages* in the *Launching the Import* section of this document on how to access these messages.

The following is a description of the API Message Table:

Field	Type	Description
Business_Object_ID	VARCHAR2(3);	Error Handling API will be shared by ECO, BOM and RTG business objects. The default ID is ECO.
Message_Text	VARCHAR2(2000)	The actual message that the user sees. Please see below for format information.

Field	Type	Description
Entity_Id	VARCHAR2(3)	The entity that this message belongs to. This may hold BO, ECO, REV, RI, RC, RD, or SC. BO - Business Object ECO - ECO Header REV - Revisions(BOM, ECO, Routing) RI - Revised Items RC - Revised Components RD - Reference Designators SC - Substitute Components BH- Bills of Material Header BC - Bills of Material Comments RTG - Routing Header OP - Routing Operations Sequence RES - Operation Resources SR - Substitute Operation Resources NWK - Operation Networks
Entity_Index	NUMBER	The index of the entity array this record belongs to.

Message formats

Expected errors and warnings: The message text contains the translated and token substituted message text. Please note that the message text may contain tokens, some of which will identify the entity instances that this message is for. The following tokens identify the several entities:

- Assembly Item Name: Assembly_Item_Name
- Alternate Routing Code: Alternate_Routing_Code

- Organization Code: Orchid
- Operation Sequence Number: Op_Seq_Number
- Operation Resource Sequence Number: Res_Seq_Number
- Schedule Sequence Number: Schedule_Seq_Number
- Substitute Resource Code: Sub_Resource_Code

Unexpected errors:

<Package Name> <Procedure/Function Name> <SQL Error Number> <SQL Error Message Text>

Other Message:

An **Other Message** is a message that is logged for all records that are affected by an error in a particular record. So if an error in a routing record will cause all it's children to error out, then the following will be logged:

- For the routing itself, the error message describing the problem.
- For all records affected by the type of error that occurred in the routing, the **other message**. This message essentially mentions the following:

1. How the error has affected this record, that is, it has been error-ed out too, with a severe or fatal error status, or that it has not been processed.
2. Which record caused this record to be affected.
3. What process flow step in the offending record caused this record to be affected.

Error Handler

The program performs all error handling and messaging through the Error Handler. It makes calls to the Error Handler when an error or warning needs to be issued. The following are the functions of the Error Handler:

- Log the error/warning messages sent to it.
- Set the return status of the record in error.
- Set the return status of other records affected by this error.

The following is the input that the Error Handler expects from the calling program:

Input	Description
Business Object	Calling program must pass the whole business object as-is.
Message and Token List	List of messages generated for error in the current record. See below for description of this list.
Error Status	Status the record in error should be set to.
Error Level	Business Object hierarchy level that current record is an instance of. That is, the entity that the record in error belongs to.
Entity Array Index	Index of record in error in its encompassing entity array. Does not apply to Routing Header.
Error Scope	Indicates depth of error, that is, how many other records are affected by it.
Other Message and Token List	Message generated for the other affected records. See below for description.
Other Status	Status the other affected records should be set to.

The calling program must trap the error and send the above details to the Error Handler. The Error Handler handles the error and returns the altered object to the calling program.

Message and Token List Records

The Message and Token List, and the Other Message and Token List are temporary arrays that the calling program maintains. They hold message-and-token records. The Error Handler must log these messages into the API Message List. The calling program may want some of these message record tokens to be translated (such tokens are typically messages themselves).

For expected errors and warnings, the translated message text is retrieved using the message name. Then, after any requested token translation is performed, the tokens are substituted into the translated message text, and the message is logged. For unexpected errors, the calling program itself sends a message text, so no message retrieval is needed. The message is logged after token translation and substitution is performed.

Field	Description
Message Name	Name of the message used to retrieve the translated message text. NULL for unexpected errors.
Message Text	Message text for unexpected errors.
Token Name	Name of the token in the message.
Token Value	Value of the token in the message.
Translate	Should this token value be translated?

Since each message may have more than one token, the Message and Token List has as many occurrences of the same message as there are tokens in it. The same applies to the Other Message and Token List, except that this list needs to carry only one message which is assigned to all other affected records. Since this lone message may have more than one token, there may be more than one occurrence of the message in the list.

Please note that the API Message List is not public and must be accessed through the Messaging API's provided to access the message list.

These are the message list API's that can be used to perform various operations on the Message List.

7

Cost Management Open Interfaces

This chapter covers the following topics:

- Periodic Cost Open Interface
- Setting Up the Interface
- Periodic Cost Open Interface Runtime Options
- Inserting into the Periodic Cost Interface Tables
- Validation
- Reviewing Failed Rows
- Cost Import Interface
- Setting Up the Cost Import Interface

Periodic Cost Open Interface

The Oracle Periodic Cost Open Interface provides an open interface for you to easily load periodic item costs from external applications or legacy systems and migrate them into the Oracle Cost Management Application. This interface should only be used to bring in periodic costs for the first opened periodic period. It cannot be used for subsequent periods. Costs in subsequent periods are calculated by the system.

See Also

Oracle Cost Management Users Guide

Oracle Bill of Materials eTechnical Reference Manual

Functional Overview

The Periodic Cost Open Interface lets you import Periodic Costing data into the Oracle Cost Management Application. You can specify just the few required attributes and let the system do validation of imported data.

Initially, Periodic Costs need to be loaded into the following interface tables with a

value of PROCESS_FLAG = 1:

- CST_PC_ITEM_COST_INTERFACE is used for the Periodic Cost data and all Periodic Cost related attributes. This is the header table storing cost information in the interface.

- CST_PC_COST_DET_INTERFACE is used for capturing the Periodic Cost details and related cost detail attributes.

If the Periodic Costs and/or the periodic cost detail rows fail any validation, the master/detail rows are flagged with errors. The columns ERROR_FLAG, ERROR_EXPLANATION, and PROCESS_FLAG are populated and the import is failed.

If the import succeeds validation, the destination tables for periodic costs are:

- CST_PAC_ITEM_COSTS

- CST_PAC_ITEM_COST_DETAILS

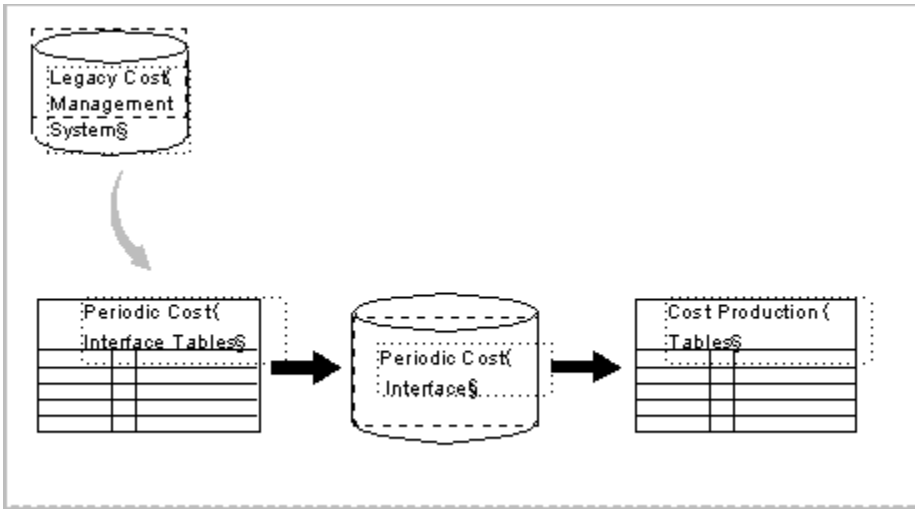
- CST_PAC_QUANTITY_LAYERS

See Also

Oracle Cost Management Users Guide

Oracle Bill of Materials eTechnical Reference Manual.

Periodic Cost Open Interface



Setting Up the Interface

Create Indexes for Performance

You should create indexes on the following columns to improve Periodic Cost Open Interface performance.

- Unique index on INTERFACE_HEADER_ID on CST_PC_ITEM_COST_INTERFACE
- Unique index on INTERFACE_LINE_ID on CST_PC_COST_DET_INTERFACE
- Non-unique index on INTERFACE_GROUP_ID column in the CST_PC_ITEM_COST_INTERFACE table
- Non-unique index on INTERFACE_HEADER_ID column in the CST_PC_COST_DET_INTERFACE table

Set Profile Option Defaults

None. All defaults are set up as part of the basic Periodic Costing set up.

Periodic Cost Open Interface Runtime Options

To run the Periodic Cost Open Interface, select Import Periodic Costs from the Periodic Cost menu.

These are runtime options for the Periodic Cost Open Interface:

Delete Interface Rows After Successful Import

- Yes Delete all the rows in the interface table.
- No Do not delete the rows in the interface.

Inserting into the Periodic Cost Interface Tables

Periodic Costs Interface Table Description

CST_PC_ITEM_COSTS_INTERFACE is used for the Periodic Cost data and all Periodic Cost related attributes.

CST_PC_ITEM_COSTS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
INTERFACE_HEADER_ID	NUMBER (sequence)	X			
INTERFACE_GROUP_ID	NUMBER		X		
COST_LAYER_ID	NUMBER		X		
PAC_PERIOD_ID	NUMBER		X		
COST_GROUP_ID	NUMBER		X		
COST_GROUP	VARCHAR2(10)	X			
COST_TYPE	VARCHAR2(10)	X			
PERIOD_NAME	VARCHAR2(15)	X			
INVENTORY_ITEM_ID	NUMBER	X			
QUANTITY_LAYER_ID	NUMBER		X		
BEGIN_LAYER_QUANTITY	NUMBER	X			
ISSUE_QUANTITY	NUMBER				
BUY_QUANTITY	NUMBER				

CST_PC_ITEM_COSTS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
MAKE_QUANTITY	NUMBER				
ITEM_COST	NUMBER		X		
MARKET_VALUE	NUMBER			X	
JUSTIFICATION	VARCHAR2(2000)			X	
BEGIN_ITEM_COST	NUMBER		X		
ITEM_BUY_COST	NUMBER		X		
ITEM_MAKE_COST	NUMBER		X		
PL_MATERIAL	NUMBER		X		
PL_MATERIAL_OVERHEAD	NUMBER		X		
PL_RESOURCE	NUMBER		X		
PL_OUTSIDE_PROCESSING	NUMBER		X		
PL_OVERHEAD	NUMBER		X		
TL_MATERIAL	NUMBER		X		

CST_PC_ITEM_COSTS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
TL_MATERIAL_OVERHEAD	NUMBER		X		
TL_RESOURCE	NUMBER		X		
TL_OUTSIDE_PROCESSING	NUMBER		X		
TL_OVERHEAD	NUMBER		X		
PL_ITEM_COST	NUMBER		X		
TL_ITEM_COST	NUMBER		X		
UNBURDENED_COST	NUMBER		X		
BURDEN_COST	NUMBER		X		
MATERIAL_COST	NUMBER		X		
MATERIAL_OVERHEAD_COST	NUMBER		X		
RESOURCE_COST	NUMBER		X		
OVERHEAD_COST	NUMBER		X		

CST_PC_ITEM_COSTS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
OUTSIDE_PROCESSING_COST	NUMBER		X		
PROCESS_FLAG	NUMBER	X			
REFERENCE	VARCHAR2(240)				
ERROR_FLAG	NUMBER				
ERROR_EXPLANATION	VARCHAR2(2000)				
LAST_UPDATE_DATE	DATE	X			
LAST_UPDATED_BY	NUMBER	X			
CREATION_DATE	DATE	X			
CREATED_BY	NUMBER	X			
LAST_UPDATE_LOGIN	NUMBER				
REQUEST_ID	NUMBER				
PROGRAM_APPLICATION_ID	NUMBER				

CST_PC_ITEM_COSTS_INTERFACE					
Column Name	**Type**	**Required**	**Derived**	**Optional**	**Reserved for Future Use**
PROGRAM_APPLICATION_DATE	DATE				
LOCK_FLAG	NUMBER				X

Periodic Cost Detail Interface Table Description

CST_PC_COST_DET_INTERFACE is used for importing the Periodic Cost details and related cost detail attributes.

CST_PC_COST_DET_INTERFACE					
Column Name	**Type**	**Required**	**Derived**	**Optional**	**Reserved for Future Use**
INTERFACE_LINE_ID	NUMBER (sequence)	X			
INTERFACE_HEADER_ID	NUMBER (FK)	X			
COST_LAYER_ID	NUMBER				
COST_ELEMENT_ID	NUMBER	X			
LEVEL_TYPE	NUMBER	X			
ITEM_COST	NUMBER	X			
ITEM_BUY_COST	NUMBER			X	
ITEM_MAKE_COST	NUMBER			X	

CST_PC_COST_DET_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
PROCESS_FLAG	NUMBER	X			
REFERENCE	VARCHAR2(240)			X	
ERROR_FLAG	NUMBER				
ERROR_EXPLANATION	VARCHAR2(2000)				
LAST_UPDATE_DATE	DATE	X			
LAST_UPDATED_BY	NUMBER	X			
CREATION_DATE	DATE	X			
CREATED_BY	NUMBER	X			
LAST_UPDATE_LOGIN	NUMBER				
REQUEST_ID	NUMBER				
PROGRAM_APPLICATION_ID	NUMBER				
PROGRAM_APPLICATION_DATE	DATE				

Note: For information about columns not discussed in the Interface

Manual, see Table and View Definitions, *Oracle Bills of Material eTechnical Reference Manual.*

Required Data

The Periodic Cost Interface uses the PROCESS_FLAG to indicate whether processing of the row succeeded or errored. When the data is loaded, the PROCESS_FLAG must be set to a value of *1* (Unprocessed). This will allow the import program to process the row for validation and import. The next table shows the required values for each interface table.

Table	Columns	Required Value
CST_PC_ITEM_COSTS_INTERFACE	INTERFACE_HEADER_ID	not null
	PERIOD_NAME	not null
	COST_GROUP	not null
	COST_TYPE	not null
	INVENTORY_ITEM_ID	not null
	BEGIN_LAYER_QUANTITY	not null
	PROCESS_FLAG	1
	Standard WHO column	
	LAST_UPDATE_DATE	
	LAST_UPDATED_BY	
	CREATION_DATE	
	CREATED_BY	
CST_PC_COST_DET_INTERFACE	INTERFACE_LINE_ID	not null
	INTERFACE_HEADER_ID	not null

Table	Columns	Required Value
	COST_ELEMENT_ID	not null
	ITEM_COST	not null
	LEVEL_TYPE	not null
	PROCESS_FLAG	not null
	Standard WHO column	not null
	LAST_UPDATE_DATE	not null
	LAST_UPDATED_BY	not null
	CREATION_DATE	not null
	CREATED_BY	not null

Derived Data

Issues arising during derivation of values for these columns are flagged as error and the row fails import.

Validation

The Open Interface program processes rows that have a PROCESS_FLAG = 1 (Unprocessed). Default values for derived columns are first updated. Then, the value of the PROCESS_FLAG is updated to 2 (Running) to continue validation of data.

The Periodic Cost Interface then validates each row. If a row in an interface table fails validation, the program sets the PROCESS_FLAG to 3 (Error) and sets the appropriate ERROR_FLAG and ERROR_EXPLANATION.

For all successfully validated rows, the interface inserts the rows into the appropriate Oracle Cost Management Periodic Cost tables.

Once the validated row is successfully inserted into the Oracle Cost Management Periodic Cost table, the PROCESS_FLAG is set to 7 (Complete). The rows in the interface that were successfully imported will be deleted if the runtime option Delete all the rows in the interface table = Yes is set.

If the rows in the interface are not deleted, users can manually delete them when necessary.

The PROCESS_FLAG column has one of 4 possible values:

- 1 - Unprocessed
- 2 - Running
- 3 - Error
- 7 - Complete

Validation and Error Handling

This table lists all the required Periodic Costing Open Cost Interface validations and the error conditions that result when these validations fail. When validations fail the PROCESS_FLAG is set to 3 denoting an error condition.

The Error Flag and Error Explanation detail the nature of the error.

Tables abbreviations are:

- CPCDI - CST_PC_COST_DET_INTERFACE
- CPICI - CST_PC_ITEM_COST_INTERFACE

Error Flag	Validation Rule	Error Explanation	Table
1	No Corresponding Header	No Corresponding Header	CPCDI
2	COST_ELEMENT_ID must be one of following: 1 - Material 2 - Material Overhead 3 - Resource 4 - Outside Processing 5 - Overhead	Invalid Cost Element	CPCDI

Error Flag	Validation Rule	Error Explanation	Table
3	LEVEL_TYPE must be either: 1 - This (current) Level 2 - Previous Level	Invalid Level Type	CPCDI
4	All values in the column COST_GROUP_ID must exist in the CST_COST_GROUP_ASSIGNMENTS table.	Invalid Cost Group or No Organizations in Cost Group	CPICI
5	All values in the COST_TYPE_ID column must exist in the CST_LE_COST_TYPES table.	Invalid Cost Type or Cost Type not in Legal Entity	CPICI
6	All values in the column PAC_PERIOD_ID must exist in the CST_PAC_PERIODS table. The period for which the data is imported (into the CST_PAC_ITEM_COSTS, CST_PAC_ITEM_COST_DETAILS and CST_PAC_QUANTITY_LAYERS tables) should have a status of OPEN and the first defined period in the CST_PAC_PERIODS table.	Period Name Invalid or period is not the first opened for the legal entity and cost type	CPICI
7	Line Record Missing	Line Record Missing	CPICI

Error Flag	Validation Rule	Error Explanation	Table
8	All values in the column INVENTORY_ITEM_ID must exist in the MTL_SYSTEM_ITEMS table. All items should belong to at least one of the organizations in the MTL_SYSTEM_ITEMS, CST_COST_GROUPS and CST_ COST_ GROUP_ASSIGNMENTS tables.	Invalid Inventory Item or item not in Periodic Costing Organization	CPICI
9	Records brought into the CST_PAC_ITEM_COSTS, CST_PAC_ITEM_COST_DETAILS and CST_PAC_QUANTITY_LAYERS tables must not pre-exist in the relevant tables at the time of import	Item Cost Exists	CPICI
10	Header Failed as Line Failed	Header Failed as Line Failed	CPICI
11	Line Failed as Header failed	Line Failed as Header failed	CPCDI
12	MAKE_QUANTITY, BUY_QUANTITY, ISSUE_QUANTITY must all be 0	QOH Value Error	CPCDI
13	All quantity and cost columns should have a value of >= 0	Cost Value Error	CPICI CPCDI

Error Flag	Validation Rule	Error Explanation	Table
14	If MARKET_VALUE is supplied, it should be less than the computed ITEM_COST value in CPICI	Market Value Error	CPICI
15	If MARKET_VALUE is supplied, then JUSTIFICATION should be NOT NULL	Justification cannot be NULL	CPICI
99	Other	Other	CPICI CPCDI

Importing Additional Periodic Cost Details

Imported cost data for derived columns will overwrite any user specified values.

Reviewing Failed Rows

You can review and report rows in any of the interface tables using SQL*Plus or any custom reports you develop. Since all rows in the interface tables have a value for PROCESS_FLAG, you can identify records that have not been successfully imported into Oracle Cost Management. If you want to reprocess any row, put the flag back to 1, and clear all errors (columns ERROR_FLAG and ERROR_EXPLANATION).

Log File Messages

Message Code	Message Text	Type
CST_PAC_OCI_START_VALIDATION	Starting to validate Periodic Open Cost Interface table information	Action
CST_PAC_OCI_ERR_CGLE	Error: Could not fetch cost group and/or legal entity identifiers	Error

Message Code	Message Text	Type
CST_PAC_OCI_SUCC_CGLE	Success: Retrieved cost group and legal entity identifiers	Success
CST_PAC_OCI_ERR_CTCM	Error: Could not fetch cost type and/or cost method identifiers	Error
CST_PAC_OCI_SUCC_CTCM	Success: Retrieved cost type and/or cost method identifiers	Success
CST_PAC_OCI_ERR_PID	Error: Could not fetch proper Periodic Costing Period identifier	Error
CST_PAC_OCI_SUCC_PID	Success: Retrieved Periodic Costing Period identifier	Success
CST_PAC_OCI_ERR_ITEMID	Error: Could not validate inventory item identifier	Error
CST_PAC_OCI_SUCC_ITEMID	Success: Validated inventory item identifier	Success
CST_PAC_OCI_ERR_COST	Error: Item cost exists in the System	Error
CST_PAC_OCI_SUCC_COST	Success: Item cost does not exist in the System	Success
CST_PAC_OCI_ERR_LAYER_EXISTS	Error: Layer Validation Error	Error
CST_PAC_OCI_ERR_LAYER_GEN	Error: Layer Creation Error	Error
CST_PAC_OCI_SUCC_LAYER_GEN	Success: Layer Creation Successful	Success
CST_PAC_OCI_UPDATE_NULL	Action: Updating all derived columns to NULL in the Interface Tables	Action

Message Code	Message Text	Type
CST_PAC_OCI_CALCULATE	Action: Computing and Updating derived columns in the Interface Tables	Action
CST_PAC_OCI_SUCC_CALCULATE	Success: Computation of all derived columns compete	Success
CST_PAC_OCI_ERR_MARKET	Error: Market Value cannot be more than computed Item Cost Value	Error
CST_PAC_OCI_ERR_JUSTIFICATION	Error: Justification cannot be NULL if Market Value is supplied	Error
CST_PAC_OCI_START_INSERT	Action: Starting to insert rows into Oracle Cost Management tables	Action
CST_PAC_OCI_SUCC_CPIC	Success: Inserted into CST_PAC_ITEM_COSTS table	Success
CST_PAC_OCI_SUCC_CPICD	Success: Inserted into CST_PAC_ITEM_COST_DETAILS table	Success
CST_PAC_OCI_SUCC_CPQL	Success: Inserted into CST_PAC_QUANTITY_LAYERS table	Success
CST_PAC_OCI_START_PURGE	Action: Starting to purge processed rows from interface tables	Action
CST_PAC_OCI_SUCC_PURGE	Success: Purge Process Complete	Success
CST_PAC_OCI_CHECK_LOG	NOTE: Please check log for error transactions	Note

Cost Import Interface

The Oracle Cost Import Interface enables you to import costs for items from legacy systems, and import new cost information for existing items. Importing resource costs and resource overhead rates is also supported. You will also be able to replace existing cost information with the new cost information. However, updating of existing costs is not supported. Importing costs into frozen cost type is also not supported. The item costs will have to be imported into another cost type and then the cost update may be run to update the frozen cost type.

Parameters and Descriptions

This section includes the details for each of the following input parameters:

- Import Cost Option
- Mode to Run This Request
- Group ID Option
- Cost Type to Import To
- Delete Successful Rows

Import Cost Option

A list of values is provided from which the user can select one of the import options which may be either to import Only item costs, Only resource costs, Only overhead rates, or All cost information. The following list contains the Option and the table from which data is processed:

- Only item cost - cst_item_cst_dtls_interface
- Only resource costs - cst_resource_costs_interface
- Only overhead rates - cst_res_overheas_interface and cst_dept_overheads_interface
- All Cost Information - From all four interface tables

If default material subelement is specified for an Organization and the material row in cst_item_cst_dtls_interface has no subelement specified against it then the org level default would be picked up for the material row.

Mode to Run This Request

A list of values is provided with possible two values:

- Insert new cost - This mode can be used if the user is importing large numbers of items and is unsure if Item/Organization/Cost Type combination exists in the production tables. If it does exist, then the row in the interface table would be flagged as an error and not imported. This would prevent any accidental overwrite of existing data.

- Remove and replace costs - All the previous cost information for this item, cost_type, organization combination will be deleted from the production tables and the new information will overwrite (replace) the existing one.

Group ID Option

A list of values is provided from which the user can either select All or Specific Group ID. If the user wishes to submit multiple Cost Import process requests they can do so by submitting one request per group id. For doing so the data in the interface tables should be stamped with distinct group id value by using the NEXTVAL from the sequence generator CST_LISTS_S. The use of this sequence generator is a must for generating multiple groups or may lead to data corruption as these interface tables are used by other processes as well.

If the user selects All from the list then a group ID generated by a sequence will replace the group ID in the interface tables (if any) and all the unprocessed rows from the four interface tables (cst_item_cst_dtls_interface, cst_resource_costs_interface, cst_res_overheads_interface, and cst_dept_overheads_interface) will be processed in one run.

Cost Type to Import To

The user is provided with a list of values from which they need to select the cost type in which they wish to import the cost information. Even if the user has populated a cost type or cost type ID in the interface tables, it would be overwritten with the one that is selected here. The cost types that the user can pick from is restricted to the multi-org, updatable cost types.

Delete Successful Rows

This parameter determines whether the successfully processed rows should be deleted from the interface tables at the end of the run. If the user selects Yes then all the successful rows will be deleted (rows that do not have their error flag set to E).

Setting Up the Cost Import Interface

The Cost Import Interface includes four tables:

- CST_RESOURCE_COSTS_INTERFACE

- CST_RES_OVERHEADS_INTERFACE

- CST_DEPT_OVERHEADS_INTERFACE
- CST_ITEM_CST_DTLS_INTERFACE

Note: CICI is not used by the Cost Import process.

Note: The Process_flag indicates what stage of validation this row is in. This will be populated as the rows are being processed in various phases. This must be set to 1 initially.

Obtaining a Trace

You need to run "exec FND_STATS.gather_table_stats('BOM','<name of the table>',10)" before the Cost Import process is run. This will ensure that the plan that is obtained, after running the process, gives a good picture of the execution plan. This statement gives the cost optimizer an idea of the number of rows that will be present in the table. If this statement is not executed before the process is run then the plan that is obtained will be based on an old estimate of the number of rows and it may not provide an accurate picture of the actual execution plan. For example, if the user is planning to run the trace for importing item costs, then they will need to run the above statement for the 3 tables CICDI,CIC,CICD before running the import process and the plan that is obtained will give a good picture of the actual execution plan.

CST_RESOURCE_COSTS_INTERFACE

CST_RESOURCE_COST_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Resource_ID	Number	X			
Cost_type_ID	Number				X
Organization_ID	Number	X			
Last_update_date	Date			X	

CST_RESOURCE_COST_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Last_updated_by	Number			X	
Creation_date	Date			X	
Created_by	Number			X	
Last_update_login	Number			X	
Resource_rate	Number	X			
Request_ID	Number			X	
Program_application_ID	Number			X	
Program_ID	Number			X	
Program_update_date	Date			X	
Attirbute_category	Varchar2(30)			X	
Error_flag	Varchar2(1)			X	
Error_code	Varchar2(240)			X	
Error_explanation	Varchar2(240)			X	
Resource_code	Varchar2(20)			X	
Cost_type	Varchar2(10)				X

CST_RESOURCE_COST_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Organization_code	Varchar2(3)			X	
Transaction_ID	Number			X	
Group_ID	Number			X	
Group_description	Varchar2(80)			X	
Process_flag	Number	X			

Column Descriptions

Resource_ID - The Resource (subelement) identifier. This must be in BOM_RESOURCES. Either resource_ID or resource_code has to be filled up.

Cost_type_ID - The cost type identifier. All of the costs will be imported against the cost type specified in the request parameter. If a cost type is provided in this column, it will be overwritten.

Organization_ID - Organization identifier. This must be in mtl_parameters. Either Organization_code or organization_id must be populated.

Last_update_date - Standard Who column.

Last_updated_by - Standard Who column.

Creation_date - Standard Who column.

Created_by - Standard Who column.

Last_update_logon - Standard Who column.

Resource_rate - Resource unit cost.

Request_ID - Concurrent Who column.

Program_applicatin_ID - Concurrent Who column.

Program_ID - Concurrent Who column.

Program_update_date - Concurrent Who column.

Attribute_category - Descriptive flexfield structure defining the column.

Error_flag - This is set to E if there is an error.

Error_code - This identifies the error code.

Error_explanation - The error explanation. This displays the cause of the error which can be reviewed using the list of common cost import error explanations listed in this guide.

Resource_code - The resource (subelement) identifier. This must be in BOM_RESOURCES.

Cost_type - The cost type identifier. This will be overwritten with the cost type mentioned while specified in the request parameters.

Organization_code - Organization identifier. This must be in MTL_PARAMETERS.

Transaction_ID - Unique row identifier. This is auto generated.

Group_ID - Batch identifier. You can populate this column if the process is to be run by group ID.

Group_description - Description of the Group_ID.

Process_flag - Flag that indicates what stage of validation this row is in. This will be populated as the rows are being processed in various phases. This must be set to 1 initially.

CST_RESOURCE_COSTS_INTERFACE Table Validations

- The Resource_ID (or Resource_code) and Organization_ID combination must have been defined. In other words no new resources will be created. If the Resource_ID, Organization_ID is not found in the BOM_RESOURCES, the rows for that particular item, organization combination will be errored out and no item costs will be imported.

- .If the Resource_ID mentioned in the CST_RESOURCE_COSTS_INTERFACE table has the Functional_currency flag (in the BOM_RESOURCES table) set to yes, then no resource rates can be entered for this Resource_ID organization combination. In other words the functional currency flag must not be equal to 1 for the resource to have a rate.

- The resource rate cannot be null. It must have some value.

- No two rows can be defined for the same resource, organization, and cost type combination.

CST_RES_OVERHEADS_INTERFACE

CST_RES_OVERHEADS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Resource_ID	Number	X			
Cost_type_ID	Number				X
Organization_ID	Number	X			
Overhead_ID	Number	X			
Last_update_date	Date			X	
Last_updated_by	Number			X	
Creation_date	Date			X	
Created_by	Number			X	
Last_update_login	Number			X	
Request_ID	Number			X	
Program_application_ID	Number			X	
Program_ID	Number			X	
Program_update_date	Date			X	
Attirbute_category	Varchar2(30)			X	
Error_code	Varchar2(240)			X	

CST_RES_OVERHEADS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Error_explanation	Varchar2(240)			X	
Resource_code	Varchar2(20)			X	
Cost_type	Varchar2(10)				X
Organization_code	Varchar2(3)			X	
Overhead	Varchar2(30)			X	
Transaction_ID	Number			X	
Group_ID	Number			X	
Group_description	Varchar2(80)			X	
Error_flag	Varchar2(1)			X	
Process_flag	Number	X			

Column Descriptions

Resource_ID - The Resource (subelement) identifier. This must be in BOM_RESOURCES. Either Resource_ID or Resource_code has to be filled up.

Cost_type_ID - The cost type identifier. All of the costs will be imported against the cost type specified in the request parameters. If a cost type is provided in this column, it will be overwritten.

Organization_ID - Organization identifier. This must be in MTL_PARAMETERS. Either Organization_code or Organization_id must be populated.

Overhead_ID - Overhead subelement identifier. This must be in BOM_RESOURCES. Either Overhead_ID or Overhead must be populated.

Last_update_date - Standard Who column.

Last_updated_by - Standard Who column.

Creation_date - Standard Who column.

Created_by - Standard Who column.

Last_update_logon - Standard Who column.

Request_ID - Concurrent Who column.

Program_applicatin_ID - Concurrent Who column.

Program_ID - Concurrent Who column.

Program_update_date - Concurrent Who column.

Attribute_category - Descriptive flexfield structure defining the column.

Error_flag - This is set to E if there is an error.

Error_code - This identifies the error code.

Error_explanation - The error explanation. This displays the cause of the error which can be reviewed using the list of common cost import error explanations listed in this guide.

Resource_code - The resource (subelement) identifier. This must be in BOM_RESOURCES.

Cost_type - The cost type identifier. This will be overwritten with the cost type specified in the request parameters.

Organization_code - Organization identifier. This must be in MTL_PARAMETERS.

Overhead - Overhead subelement identifier.

Transaction_ID - Unique row identifier. This is auto generated.

Group_ID - Batch identifier. You can populate this column if the process is to be run by group ID.

Group_description - Description of the Group_ID.

Process_flag - Flag that indicates what stage of validation this row is in. This will be populated as the rows are being processed in various phases. This must be set to 1 initially.

CST_RES_OVERHEADS_INTERFACE Table Validations

- The Organization_ID, Resource_ID combination must exist in the BOM_RESOURCES. No new resources will be created.

- The resource_ID or the Resource_code must be provided and must be valid for that Organization_ID. If both are provided then they must match.

- The Organization_ID and Overhead_ID combination must exist in the BOM_RESOURCES table. The Overhead_ID or the Overhead has to be provided.

CST_DEPT_OVERHEADS_INTERFACE

CST_DEPT_OVERHEADS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Department_ID	Number	X			
Cost_type_ID	Number				X
Organization_ID	Number	X			
Overhead_ID	Number	X			
Last_update_date	Date			X	
Last_updated_by	Number			X	
Creation_date	Date			X	
Created_by	Number			X	
Last_update_login	Number			X	
Basis_type	Number			X	
Rate_or_amount	Number	X			
Activity_ID	Number			X	
Request_ID	Number			X	
Program_application_ID	Number			X	

CST_DEPT_OVERHEADS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Program_ID	Number			X	
Program_update_date	Date			X	
Attirbute_category	Varchar2(30)			X	
Error_code	Varchar2(240)			X	
Error_explanation	Varchar2(240)			X	
Department_code	Number			X	
Cost_type	Varchar2(10)				X
Organization_code	Varchar2(3)			X	
Overhead	Varchar2(30)			X	
Transaction_ID	Number			X	
Group_ID	Number			X	
Group_description	Varchar2(80)			X	
Process_flag	Number	X			
Activity	Varchar2(20)			X	
Error_flag	Varchar2(1)			X	

Column Descriptions

Department_ID - Department identifier. This must be in BOM_DEPARTMENTS. Either Department_ID or Department must be populated.

Cost_type_ID - The cost type identifier. All of the costs will be imported against the cost type specified in the request parameters. If a cost type is provided in this column, it will be overwritten.

Organization_ID - Organization identifier. This must be in MTL_PARAMETERS. Either Organization_code or Organization_id must be populated.

Overhead_ID - Overhead subelement identifier. This must be in BOM_RESOURCES. Either Overhead_ID or Overhead must be populated.

Last_update_date - Standard Who column.

Last_updated_by - Standard Who column.

Creation_date - Standard Who column.

Created_by - Standard Who column.

Last_update_logon - Standard Who column.

Basis_type - The basis type identifier. This must be between 1 and 6.

Rate_or_amount - Rate or amount for the overhead subelement. You must populate this column.

Activity_ID - Activity identifier. This must be in CST_ACTIVITIES.

Request_ID - Concurrent Who column.

Program_applicatin_ID - Concurrent Who column.

Program_ID - Concurrent Who column.

Program_update_date - Concurrent Who column.

Attribute_category - Descriptive flexfield structure defining the column.

Error_flag - This is set to E if there is an error.

Error_code - This identifies the error code.

Error_explanation - The error explanation. This displays the cause of the error which can be reviewed using the list of common cost import error explanations further in this guide.

Department - Department identifier.

Cost_type - Cost type identifier. This must be in CST_COST_TYPES. This will be overwritten with the cost type specified in the request parameters.

Organization_code - Organization identifier. This must be in MTL_PARAMETERS.

Overhead - Overhead subelement identifier.

Transaction_ID - Unique row identifier. This is auto generated.

Group_ID - Batch identifier. You can also populate this column if the process is to be run by group ID.

Group_description - Description of the Group_ID.

Process_flag - Flag that indicates what stage of validation this row is in. This will be populated as the rows are being processed in various phases. This must be set to 1 initially.

Activity - Activity identifier. This must be in CST_ACTIVITIES.

CST_DEPT_OVERHEADS_INTERFACE Table Validations

- Either Department_ID or Department will have to be provided. If both of them are provided and the Department_ID does not match Department, the row will error out. If only the Department is provided, the Department_ID column will be populated from the BOM_DEPARTMENTS table.
- Either Overhead_ID or Overhead will have to be provided wherever necessary. If both of them are provided and the Overhead_ID does not match Overhead, the row will error out. If only the Overhead is provided, the Overhead_ID column will be populated from the BOM_RESOURCES table.
- The Overhead_ID, Organization_ID combination that is provided in the CST_DEPT_OVERHEADS_INTERFACE table must be defined in the BOM_RESOURCES table for the Cost_element_ID=5 (cost element overhead).
- The Basis_type must be a valid number between 1 and 6.

CST_ITEM_CST_DTLS_INTERFACE

CST_ITEM_CST_DTLS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Cost_type_ID	Number				X
Cost_type	Varchar2(10)				X
Last_update_ date	Date			X	

CST_ITEM_CST_DTLS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Last_updated_by	Number			X	
Creation_date	Date			X	
Created_by	Number			X	
Last_update_login	Number			X	
Organization_ID	Number	X			
Organization_code	Varchar2(3)			X	
Operation_sequence_ID	Number			X	
Operation_seq_num	Number			X	
Department_ID	Number			X	
Department	Varchar2(10)			X	
Level_type	Number			X	
Activity_ID	Number			X	
Activity	Varchar2(10)			X	
Resource_seq_num	Number			X	
Resource_ID	Number	X			

CST_ITEM_CST_DTLS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Resource_code	Varchar2(10)			X	
Rollup_source_type	Number			X	
Activity_context	Varchar2(30)			X	
Request_ID	Number			X	
Item_units	Number			X	
Activity_units	Number			X	
Basis_type	Number			X	
Basis_resource_ID	Number			X	
Basis_resource_code	Varchar2(10)			X	
Usage_rate_or_amount	Number	X			
Basis_factor	Number			X	
Resource_rate	Number			X	
Net_yield_or_shrinkage_factor	Number			X	
Item_cost	Number			X	

CST_ITEM_CST_DTLS_INTERFACE

Column Name	Type	Required	Derived	Optional	Reserved for Future Use
Program_application_ID	Number			X	
Program_ID	Number			X	
Program_update_date	Date			X	
Attribute_category	Varchar2(30)			X	
Lot_size	Number			X	
Based_on_rollup_flag	Number			X	
Shrinkage_rate	Number			X	
Inventory_asset_flag	Number			X	
Cost_element_ID	Number	X			
Cost_element	Varchar2(50)			X	
Item_number	Varchar2(81)			X	
Group_ID	Number			X	
Group_description	Varchar2(80)			X	
Transaction_ID	Number			X	
Error_flag	Varchar2(1)			X	

CST_ITEM_CST_DTLS_INTERFACE					
Column Name	**Type**	**Required**	**Derived**	**Optional**	**Reserved for Future Use**
Error_code	Varchar2(240)			X	
Error_explanation	Varchar2(240)			X	
Process_flag	Number	X			

Column Descriptions

Inventory_item_ID - Item identifier. This must be in MTL_SYSTEM_ITEMS.

Cost_type_ID - Cost type identifier. This will be overwritten with the cost type provided while submitting the request.

Cost_type - Cost type identifier. This will be overwritten with the cost type provided while submitting the request.

Last_update_date - Standard Who column.

Last_updated_by - Standard Who column.

Creation_date - Standard Who column.

Created_by - Standard Who column.

Last_update_login - Standard Who column.

Organization_ID - Organization identifier. This must be in MTL_PARAMETERS. Either Organization_ID or Organization_code must be specified.

Organization_code - Organization identifier. Must be in MTL_PARAMETERS.

Operation_sequence_ID - Operation sequence identifier. This is not used.

Operation_seq_num - Operation sequence number in a routing. This is not used.

Department_ID - Not used.

Department - Not used.

Level_type - Level at which costs are incurred. Only 1 is supported.

Activity_ID - Activity identifier. This must be in CST_ACTIVITIES.

Activity - Activity identifier. This must be in CST_ACTIVITIES. If Activity is mentioned, then Activity_ID is populated from CST_ACTIVITIES. Otherwise the default is null.

Resource_seq_num - Not used.

Resource_ID - Resource (subelement) identifier. This must be in BOM_RESOURCES. Either of Resource_ID or Resource_code must be entered for any cost elements other than material. This is either defined by the user or populated from BOM_RESOURCES based on the Resource_code.

Recource_code - Resource (subelement) identifier. This must be in BOM_RESOURCES.

Rollup_source_type - Cost source. This must be 1, which is user defined.

Activity_context - Column for defining activity unit information.

Request_ID - Concurrent Who column.

Item_units - Number of units the activity cost is applied to. This must be greater than 0.

Activity_units - Number of activity units applied to the item costs. This must be greater than, or equal to, 0.

Basis_type - Basis type identifier. This must be between 1 and 6.

Basis_resource_ID - Not used.

Basis_resource_code - Not used.

Usage_rate_or_amount - Number of resource units, overhead rate, or activity unit cost per basis. This must be greater than 0.

Basis_factor - Basis factor. This is calculated from the Basis_type and lot size.

Resource_rate - Resource unit cost. This must be null.

Net_yield_or_shrinkage_factor - Item shrinkage factor. This is computed from the item shrinkage factor.

Item_cost - Computed cost of the item. This is the product of Basis_factor, Resource_rate, Usage_rate_or_amount, Net_yield_or_shrinkage_rate.

Program_application_ID - Concurrent Who column.

Program_ID - Concurrent Who column.

Program_update_date - Concurrent Who column.

Attribute_category - Descriptive flexfield structure defining column.

Lot_size - Lot size. This must be greater than 0 if specified.

Based_on_rollup_flag - Flag that indicates whether the cost has to be rolled up. This must be 1 or 2.

Shrinkage_rate - Manufacturing shrinkage rate (for make items only). This must be between 0 and 0.99 (must be less than 1).

Inventory_asset_flag - Flag that indicates whether the item is asset or expense item. This must be 1 for it to be able to have any costs.

Cost_element_ID - Cost element identifier. This must be in CST_COST_ELEMENTS. Either of the Cost_element_ID or Cost_element has to be populated.

Cost_element - Cost element identifier. This must be in CST_COST_ELEMENTS.

Item_number - Not used.

Group_ID - Batch identifier. You can also populate this column if the process is to be run by group ID.

Group_description - Description of the Group_ID.

Transaction_ID - Unique row identifier. This is populated by a sequence.

Error_flag - This is set to E if there is an error.

Error_code - The error code that identifies the error.

Error_explanation - The error explanation. This displays the cause of the error which can be reviewed using the list of common cost import error explanations further in this guide.

Process_flag - Flag that indicates what stage of validation this row is in. This will be populated as the rows are being processed in various phases. This must be set to 1 initially.

CST_ITEM_CST_DTLS_INTERFACE Table Validations

- The Organization_ID, Cost_element_ID, Resource_ID combination in the CICDI must be defined in BOM_RESOURCES. No new resources will be supported. Also, subelements with Functional_currency_code set to 1 only will be supported for resource and outside processing Cost_elements.

- The Resource_ID or Resource_code has to be provided for cost elements other than material. For material cost element, if the Resource and Resource_ID is not provided, it will be defaulted from MTL_PARAMETERS for this organization.

- Only subelements with functional currency flag set to 1 will be allowed to be imported for cost elements resource and outside processing.

- If Activity is provided, the Activity_ID will be populated from CST_ACTIVITIES. If Activity_ID is provided, the Organization_ID, Activity_ID combination must exist in the CST_ACTIVITIES table. If both Activity_ID and Activity is provided, then the Activity_ID should match the Activity.

- If any of the activity related information like the activity units, activity context, etc. is provided, it will be inserted into CST_ITEM_COSTS/CST_ITEM_COST_DETAILS without any validation.

- The Cost_element or Cost_element_id must be provided. If both are provided then they should match.

- The flags Based_on_rollup_flag, Inventory_asset_flag, and the columns Lot_size, Shrinkage_rate should be the same for all rows that correspond to a particular

Inventory_item, Organization, Cost_type combination. If they do not match (are not the same for the same Inventory_item, Organization, Cost_type combination) then all the rows for that item will be errored out and no costs for that item, organization, cost type combination will be imported. If they are left empty, they will be defaulted from the item/organization/default cost type combination where the default cost type is derived from the CST_COST_TYPES table for the provided cost type.

- The Level_type is 1 referring to This level cost information. If any other level type is mentioned, all the rows for that item, organization, cost type combination will error out.
- The Usage_rate_or_amount cannot be null.
- The Rollup_source_type must always be 1 (user defined costs only).
- The Resource_rate must always be null.
- If the item is not an inventory asset item (i.e. the inventory asset flat is set to 2 in the CICDI or if the inventory_asset_flag is set to N in MTL_SYSTEM_ITEMS for this item, organization combination) then no costs will be imported.
- Basis types of only 1 and 2 are allowed for any cost element other than material overhead. For material overhead all basis types are allowed. If the basis type of Activity is provided, then there have to be both activity units and item units defined. Moreover, Item_units_ cannot be zero if the basis type is Activity.
- Shrinkage_rate can never exceed 1.
- Lot_size can never be 0.
- If Based_on_rollup_flag is 2, then the item cannot have shrinkage rate (i.e. shrinkage rate must be 0).

Common Validations

Because of dependencies, tables CST_RESOURCE_COSTS_INTERFACE, CST_RES_OVERHEADS_INTERFACE, CST_DEPT_OVERHEADS_INTERFACE, and CST_ITEM_CST_DTLS_INTERFACE must be populated in the same order that they are listed here.

- You will have to provide either Organization_ID or Organization_code wherever required. If both are provided and if the Organization_ID does not match the Organization code, then the row will be marked error. If only the Organization_code is provided, Organization_ID will be populated from the MTL_PARAMETERS table.

- You will have to provide the Inventory_item_ID. The Inventory_item_ID and the Organization_ID must exist in the MTL_SYSTEM_ITEMS table. Only the item ID is allowed. No item names are allowed. The Item_name field is not considered even if it is populated. Only item ID is considered.

- The Group_ID specifies the batch that the row will be processed in. This is automatically populated by a sequence. All rows with that particular batch ID will then be processed in that particular run. If you decides to specify this Group_ID, then this column must be populated with the next value from the sequence CST_LISTS_S. This is necessary to avoid these rows from getting picked up by the cost copy process.

- If the organization for which the cost/rates are being imported is not an organization that can control costs (i.e. it could be an organization whose costs are controlled by another master organization), then all the rows with this Organization_ID will error out.

Error Handling and Resubmission

The following list should be verified if any of the rows contain errors during validation:

- Error_flag - This will be set to E to indicate that an error has occurred.

- Error_code - A code that will indicate the error.

- Error_explination - The actual short message that will enable the user to understand what exactly caused the error.

- Process_flag - This will indicate in what phase of the validation the validation failed.

If any row contains an error, the user can find out the reason for the error and fix the row(s) in the corresponding interface table(s). To fix a particular row, the user can use Transaction_ID to identify a particular row. To resubmit an error row for re-processing after correcting the error the user would have to set the following:

- Error_flag = null

- Process_flag = 1

The user will have to do this for all the rows for an item, organization, cost_type combination that has an error. This is because no partial cost information for an item will be imported. If there is an error with any one row for a particular item, organization, cost type combination, then all the rows for this item, organization, cost type combination for the current batch id will be contain errors.

Common Cost Import Interface Errors

This is a list of common errors that you may encounter when the rows are being validated. The error_code can be obtained from the Error_code column in the interface table. There is also a very brief explanation of the error in the Error_explanation column. A more detailed explanation of the errors is listed here:

CST_NULL_ORGANIZATION - The row that has been marked with this error is missing values for both the Organization_ID and the Organization_code columns. At least one of the two must be provided for the row to be processed.

CST_INVALID_ORGANIZATION - The row that has been marked with this error could have any of the following listed errors.

- Organization_ID that does not match any Organization_ID in the MTL_PARAMETERS table
- Organization_code that is provided does not match any Organization_code in the MTL_PARAMETERS table
- Both the Organization_ID and the Organization code that has been provided do not match (i.e the Organization_ID is not of the Organization_code that is mentioned)

CST_NULL_ITEMID - The row that has been marked with this error has missing Inventory_item_ID value. The Inventory_item_ID must be provided.

CST_NULL_COSTELEMENT - The row that has been marked with this error has missing Cost_element_ID and Cost_element. At least one of the two must be provided.

CST_INVALID_ROLLUP_SRC_TYPE - A row that has been marked with this error has an invalid value for the Rollup_source_type column. The only allowed value is 1 (user defined costs).

CST_INVALID_ITEMID - A row that has been marked with this error has an Inventory_item_ID, Organization_ID combination that does not exist (not defined in the MTL_SYSTEM_ITEMS table).

CST_INVALID_COSTELEMENT - The row that has been marked with this error has any of the following errors:

- The Cost_element_ID mentioned is not defined in the CST_COST_ELEMENTS table
- The Cost_element_ID, Cost_element combination that is mentioned does not exist in the CST_COST_ELEMENTS table

CST_INVALID_SUBELEMENT - A row that has been marked with this error has any of the following errors:

- An invalid Organization_ID, Cost_element_ID, Resource_ID combination which is not defined in the BOM_RESOURCES table

- An invalid Organization_ID, Cost_element_ID, Resource_code combination which is not defined in the BOM_RESOURCES table

- An invalid Organization_ID, Cost_element_ID, Resource_ID, Resource_code combination which is not defined in the BOM_RESOURCES table

CST_NULL_SUBELEMENT - A row that has been marked with this error indicates that the row is missing values for Resource_ID and Resource_code columns. At least one of them will have to be provided.

CST_NULL_DEFSUBELEMENT - A row that has been marked with this error suggests that it has missing values for both the Resource_ID and Resource_code columns. This error arises when either of the following conditions is true:

- You have not provided either the Resource_ID or Resource_code for a cost element other than material or material overhead

- The default subelements from MTL_PARAMETERS are null for cost element material

CST_INVALID_FUNCCODE - A row that has been marked with this error indicates that the Resource_ID mentioned does not have the Functional_currency_flag set to 1 in the BOM_RESOURCES table. Subelements with Functional_currency_code set to 1, will only be supported for resource and outside processing Cost_elements. You can check the Functional_currency_flag in the BOM_RESOURCES table for the subelement that is defined in this row.

CST_INVALID_LEVELTYPE - A row that has been marked with this error indicates that an invalid value has been entered for the Level_type column. Only Level_type value of 1 (this level costs) is supported.

CST_INVALID_CIC_FLAGS - The 4 columns Based on_rollup_flag, Inventory_asset_flag, Shrinkage_rate, and Lot_size must have the same values for all the rows for a particular Inventory_item_ID, Organization, Cost_type combination. The rows that have been marked with this error indicate that the 4 columns do not match for these rows. You have to check up the valid combination and then fix all the other rows with this value.

CST_INVALID_ROWS - If there is an error with any one row, all the rows with the same Item, Organization and Cost_type combination as this errored out row will be marked error. This is necessary to avoid importing partial costs for an item. You must fix the errored out row and then resubmit all the rows for processing.

CST_NO_ITORACUNITS - A row errored out with this error has a basis type of 6 (Activity) but either or both of Item_units and Activity_units is null. To fix this, you can mention the Item_units and the Activity_units, then resubmit for processing.

CST_ZERO_LOTSIZE - A row errored out with this error has 0 lot size. You will have to enter a non zero lot size to fix this row.

CST_NOT_COSTINGORG - A row is errored out with this error when the

Organization_ID mentioned in this row is not an organization that can control costs (i.e this organization receives costing information from another master org).

CST_NULL_DEPARTMENT - A row marked with this error has a null Department and Department_ID. To fix this, either a Department_ID or Department or both will have to be provided.

CST_CANT_INSERT - Rows are errored out with this error when the cost import process is being run in the insert new cost information only mode and there are rows that already exist for the same item/organization/cost type combination as the rows in the interface table. Since the cost information already exists, rows in the interface table will be errored out. To fix this problem, the rows will have to resubmitted with a remove and replace cost information option.

CST_DUPL_ROWS - Rows in the interface table are marked with this error when there are multiple rows in the interface table for the same Organization, Resource_ID, and Cost type combination. This error is applicable to only CST_RESOURCE_COSTS_INTERFACE, CST_RES_OVERHEADS_INTERFACE, and CST_DEPT_OVERHEADS_INTERFACE tables. To fix this error, you delete the duplicate rows.

CST_INVALID_ACTIVITY - Rows marked with this error suggest one of the following:

- The Activity_ID or Activity mentioned is invalid (does not exist in CST_ACTIVITES table)
- The Activity_ID and Activity combination is invalid
- The Activity_ID, Organization_ID combination is invalid

To fix the problem, you insert a valid Activity_ID or Activity name from CST_ACTIVITIES.

CST_INVALID_BASEDONRLP - Rows marked with this error suggest that the rows have an invalid Based_on_rollup_flag value. The only valid values are 1(yes) and 2(no).

CST_INVALID_BASISTYPE - Rows errored out with this error could have been errored out because of either of the following reasons:

- The Basis_type value is not between 1 and 6
- If this error is for a row in CST_DEPT_OVERHEADS_INTERFACE table, the Basis_type is not between 1 and 4 for cost element overhead
- The Cost_element_ID is not 2 (Material Overhead) and the Basis_type exceeds 2. For all other cost elements, other than material overhead, the only valid values are 1 and 2. For material overhead, all values (between 1 and 6) are allowed

CST_INVALID_BUYITEM - The rows marked with this error suggest that the item is not a buy item (Based_on_rollup is not 1) but a Shrinkage_rate (greater than 0) is being defined for this item. To fix this error, you can either input a value of 1 for the

Based_on_rollup_flag or make the Shrinkage_rate equal to 0.

CST_INVALID_OVERHEAD - Rows are marked with this error when either of the following hold true:

- The Overhead_ID or Overhead is invalid (does not exist in BOM_RESOURCES)
- The Overhead and Organization combination does not exist in BOM_RESOURCES
- The Overhead_ID and Overhead combination is invalid

CST_INVALID_RESRATE - This error is applicable to the rows only in CST_ITEM_CST_DTLS_INTERFACE table. The rows marked with this error suggest that the Resource_rate column does not have null as the value. Only null values are allowed.

CST_INVALID_ROLLUP_SRC_TYPE - The rows marked with this error have an invalid value of Rollup_source_type flag. It must be 1 (user defined). Only a value of 1 is supported.

CST_INVALID_SHRRATE - Rows marked with this error do not have a value of Shrinkage_rate between 0 and 1. The Shrinkage_rate can never be 1 or exceed 1.

CST_NOT_INVASSITEM - Rows are marked with this error when either of the following is true:

- The Inventory_asset_item flag is not 1 for this row in the interface table
- The Inventory_asset_item is set to N in the MTL_SYSTEM_ITEMS table for this item/organization combination

CST_INVALID_DEFCSTTYPE - Rows are marked with this error when either of Based_on_rollup_flag or Shrinkage_rate or Lot_size or Inventory_asset_flag has null values and there are no rows that exist in CST_ITEM_COSTS for this Item, Organization, Default_cost_type combination. The Default_cost_type is derived from CST_COST_TYPES for the cost type provided during the request submission. You have two options to correct this:

- Supply all the required defaults (i.e. Based_on_rollup_flag, Inventory_asset_item_flag, Shrinkage_rate, and Lot_size values)
- Rerun the request with a cost type that has a default cost type such that a row for the Item/Organization/Cost_type exists in CST_ITEM_COSTS from which defaults can be derived

CST_EXP_SUBELEMENT - Rows are marked with this error when either of the following is true:

- The subelement has expired
- The Allow_costS_flag for this subelement is set to 2 (No)

CST_REQ_ERROR - This error occurs when two requests have the same Group_ID parameter.

8

Engineering Open Interfaces

This chapter covers the following topics:

- Overview
- Features
- Business Object APIs
- Entity Process Flows
- ECO Headers
- ECO Revisions
- Revised Items
- Revised Components
- Reference Designators
- Substitute Components
- New API Packages
- Launching the Import
- Package Interaction
- Import Error Handling and Messaging
- API Messaging
- Error Handler

Overview

The Engineering Change Order (ECO) Business Object Interface enables you to import your change order information from a legacy or product data management (PDM) system into Oracle Engineering. You can create, update and delete ECO information. With the ECO Business Object Interface, you can automatically import data from your PDM or Legacy system without inserting cryptic Ids or system specific information. The

ECO Business Object Interface will import the information within an ECO synchronously while still enabling you to import more than one ECO simultaneously.

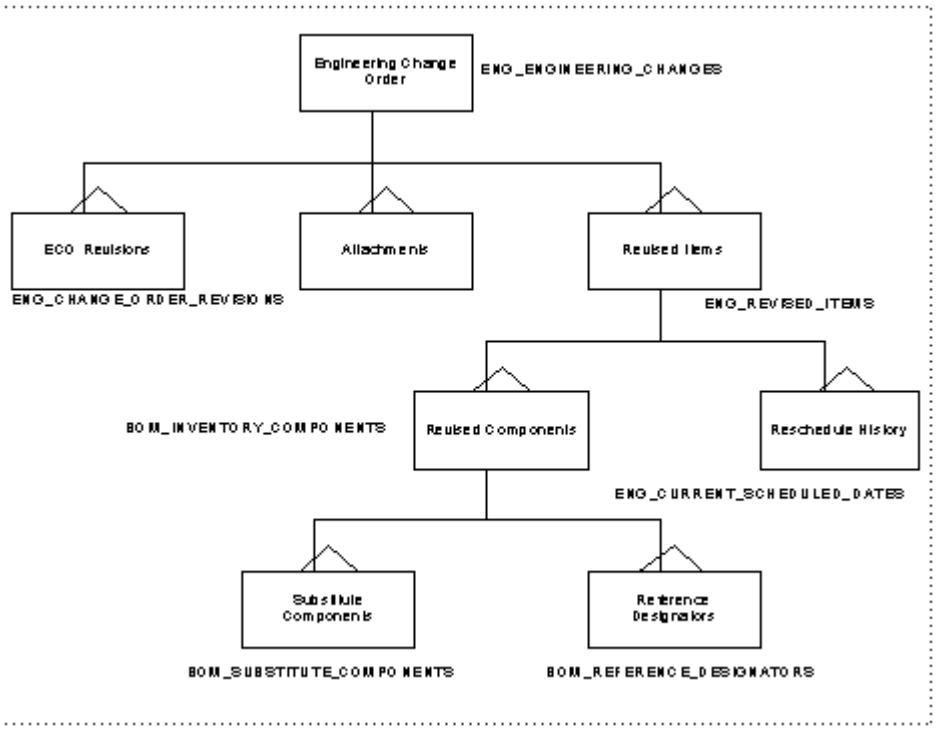
This document describes the basic business needs, major features, business object architecture and components for the Insert, Update and Delete features for the ECO Business Object Interface. Topics included are:

- Features
- Business Object APIs
- Entity Process Flows

Features

- The ECO Business Object Interface allows users to create new ECOs in the Oracle Engineering tables. The Open Interface program validates all data before importing it into the production tables. It allows users to update and delete existing ECO data.
- You can easily import create, update, and delete data. Instead of forcing you to enter cryptic ID and system specific values, the ECO Business Object Interface allows you to enter only the necessary business information that defines your ECO. The Open Interface program figures out all the remaining information.
- You should be able to create, update, and delete ECO information synchronously. The ECO Business Object Interface should process parent information before its processes child information.

ECO Information



- You should be able to process ECOs asynchronously. The ECO Business Object Interface should allows you to import different ECOs simultaneously.

ECO Business Object

The ECO business object encompasses several entities to fully model the flow of information through it. Each new ECO that is setup and processed by the user can be thought of as a business object instance.

ECO Entity Diagram

The following diagram is a representation of the ECO business object. It shows all the entities that make up the business object.

Following from the above definition of a business object, and the ECO entity diagram, each ECO business object instance contains the following specific information about its entities:

- ECO header : This entity represents general ECO attributes, such as, the ECO name, the change order type, its requestor, the organization responsible for it, etc.

- ECO Revisions : This provides additional information about the ECO. In particular,

it holds revision information for an ECO. The user may wish to keep this entity up-to-date if he/she is particular about implementing a revision control system.

- Other entities : The other entities represent detailed information about the ECO.

These entities are governed by business rules that dictate the relationships between the entities, and consequently, the processing algorithms used.

The Business Object in the Database

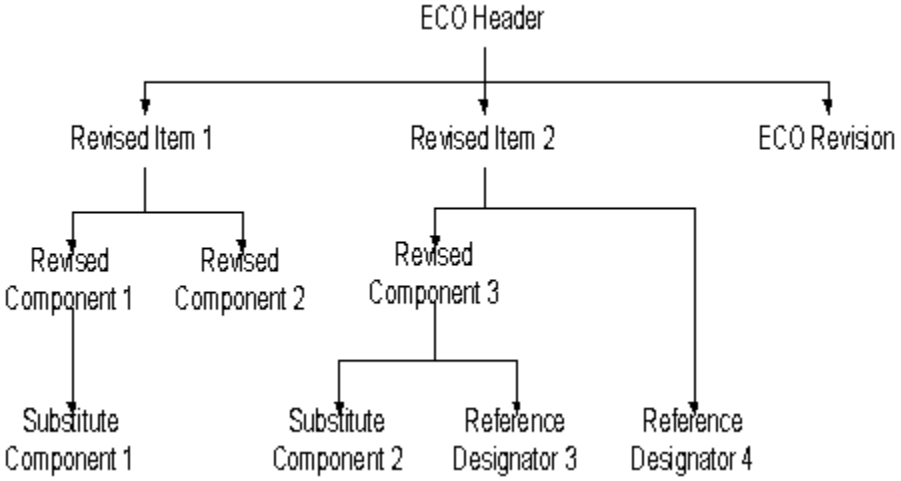
Each of the entities shown in the ECO entity diagram maps to a corresponding table in the database. The following are the existing production tables that exist, and the information they store.

Production Table	Description
ENG_ENGINEERING_CHANGES	Stores information about ECO headers
ENG_CHANGE_ORDER_REVISIONS	Stores information about revisions of an ECO
ENG_REVISED_ITEMS	Stores information about revised items on an ECO
BOM_INVENTORY_COMPONENTS	Stores information about the un-implemented single-level BOM components
BOM_REFERENCE_DESIGNATORS	Stores information about the un-implemented BOM component reference designators
BOM_SUBSTITUTE_COMPONENTS	Stores information about the un-implemented substitute components associated with a BOM component

Business Object Interface Design

The Business object architecture is a hierarchy of entities. It can also be viewed as a tree with branches, where the ECO Header entity is the parent, and each of its branches consists of child nodes (entities). Each node in the branch in turn is a parent of the child nodes under it, hence each node has a branch of its own. Here is an example:

Business Object Interface Design



ECO Business Object Architecture

The ECO Business Object Architecture is based on the business definition of an engineering change order. To import ECO information into Oracle Engineering, you only need to know the basic ECO hierarchical tree. As in a genealogical tree, the entity at the top is the parent. The entities connected directly below are its children.

Engineering Change Order: The ECO header entity is the parent and defines the basic business object. There can only be one ECO header record per business object. This entity contains information about the status, approval status, reason, priority and if it is linked to a workflow. To identify an engineering change order, you need only to specify the change notice (ECO name) and the organization in which it exists.

ECO Revisions: The revisions entity is a direct child of the ECO header. It cannot exist without an ECO. This entity stores the eco revision information. To identify an ECO Revision, you must specify the change notice, the organization in which it exists, and the name of the revision.

Revised Items: The revised items entity is also a direct child of the ECO header. It cannot exist without an ECO. You can enter information into this entity if you wish to revised an item or its bill. To identify a revised item, you must specify the change notice, the organization in which it exists, the revised item number, its effectivity date, and if there is a new item revision.

Revised Components: The revised components entity is a direct child of the revised items entity. It cannot exist without a revised item. You can enter information into this entity if you wish to add, change, or disable a component on your revised item's bill. To identify a revised component, you must specify the change notice, the organization which it exists, the revised item number, its effectivity date, if there is a new revised item revision, if you wish to revised the main bill or an alternate, the component item number, and its operation sequence number.

Substitute Component: The substitute components entity is a direct child of the revised components entity. It cannot exists without a revised component. You can enter information into this entity if you wish to add or disable a substitute component on your revised component. To identify a substitute component, you must specify the change notice, the organization in which it exists, the revised item number, its effectivity date, if there is a new revised item revision, if you wish to revised the main bill or an alternate, the component item number, its operation sequence number, the substitute component number, and whether you wish to add or disable your substitute component.

Reference Designator: The reference designator entity is a direct child of the revised components entity. It cannot exist without a revised component. You can enter information into this entity if you wish to add or disable a reference designator on your revised component. To identify a reference designator, you must specify the change notice, the organization in which it exists, your revised item number, its effectivity date, if there is a new revised item revision, if you wish to revise the main bill or an alternate, the component item number, its operation sequence number, your reference designator, and whether you wish to add or disable your reference designator.

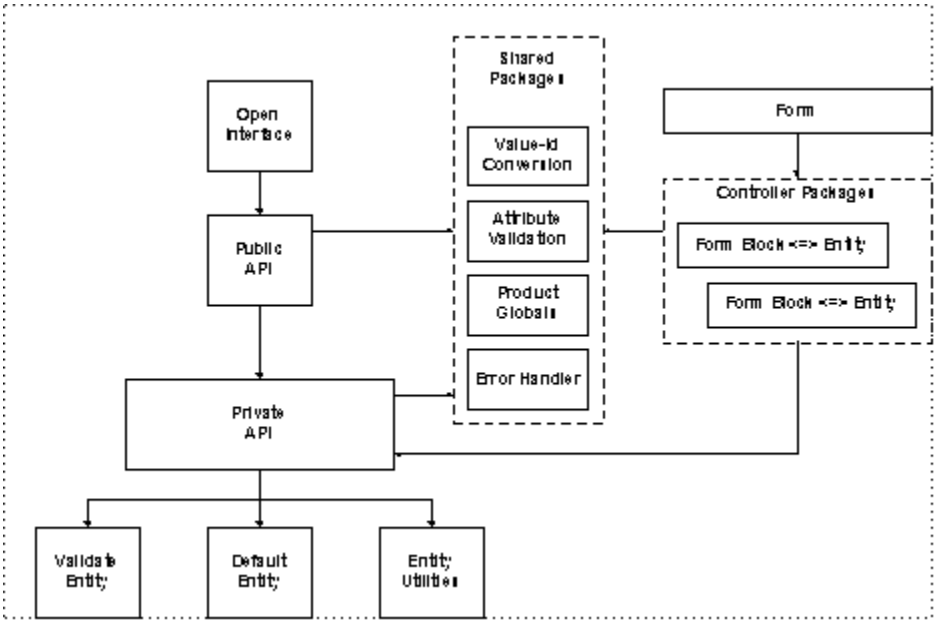
To import data into the ECO Business Object Interface, you simply need to categorize your data into ECO Business Object and identify which records you wish to import. No elaborate Ids or system specific information is required. The information you need to pass to the ECO Business Object Interface is the same information you would need to convey to any other person about your ECO.

Business Object APIs

The Business Object API framework is designed to provide flexible usage, but with a unified programming style. Business Object APIs can be called from Oracle Forms, Business object interfaces, Web-based applications, PL/SQL programs, and almost all calls that support PL/SQL calls to Oracle RDBMS. They perform a complete transaction on a business object, and always leave the database in a consistent state, regardless of the outcome of the transaction.

The APIs encompass all the business rules required to process incoming data. In a nutshell, they are the interface between the user and the database.

Business Object APIs



Business Object API Framework

The below framework demonstrates how forms and business object interfaces can use the business object API framework simultaneously.

Process Flow

Step	Purpose	Description	Error
Step 1: Pass Business Object to Public API	The program will try to import it.	Caller must pass one business object at a time. There should be only one Header record. There may be more than one record for other entities.	-N/A-

Step	Purpose	Description	Error
Step 2: Check for Organization / Change Notice uniformity	Each instance of the business object represents a particular ECO. We must ensure that all records in a business object belong to the same ECO.	All records must have the same Change Notice and Organization values. Derive Organization_Id from Organization_Code Store ECO_Name and Organization_Id value in System_Information record.	Severe Error I
Step 3: Save system information	Saves system-specific information in System_Information record since it is common to the whole business object. This information is stored in the database along with the record	Initialize User_Id, Login_Id, Prog_Appid, Prog_Id in System_Information record. Pull in values of profiles ENG: Standard Item Access, ENG: Model Item Access and ENG: Planning Item Access into STD_Item_Access, MDL_Item_Access and PLN_Item_Access respectively.	Quit importof business object
Step 4: Pick up highest level un-processed record	The import program processes a parent and all its direct and indirect children, before moving onto to a sibling. So, the highest level parent must be chosen.	The highest level record with a {return status = NULL} is picked.· When there are no more records, the program exits and transfers control to the caller.	-N/A-

Step	Purpose	Description	Error
Step 5: Convert user-unique index to unique index	Unique index helps uniquely identify a record in the database, and may consist of more than one column. User-unique index is a user-friendly equivalent of the unique index. It serves the following purposes: The user need not enter cryptic Ids If a user unique index could not be derived for a parent, its entire lineage is error-ed out since members of the lineage are referencing an invalid parent.	Derive unique index columns from user-unique index columns.	Severe Error III
Step 6: Existence Verification	The record being updated or deleted in the database must already exist. But a record being created must not. Such an error in a record must cause all children to error out, since they are referencing an invalid parent.	For CREATE, the record must not already exist. For UPDATE and DELETE, the record must exist. Query up database record into the associated OLD record.	Severe Error III

Step	Purpose	Description	Error
Step 7: Check Lineage	We must ensure that the linkage of records is correct in the business object. That is, child records must reference valid parents. A valid parent is one that exists, and is truly the current record's parent in the database.	Perform lineage checks for entity records that do not belong to the top-most entity in the hierarchy, based on Transaction_Type and the following factors: Immediate parent being referenced exists in the database, and, for UPDATE and DELETE, is truly the parent of this record in the database, OR If there is no immediate parent record in the business object, the indirect parent being referenced exists and is really the parent of the current record's parent in the database.	Severe Error III
Step 8(a): Check ECO operability	An ECO and its constituents cannot be operated upon if the ECO has already been implemented/canceled, or if the ECO is in a workflow process (with Approval _Status = Approval Requested).	First check if System_Information record has this information. If not, find it in ECO Header record in database, and set System_Information flags accordingly.	Fatal Error I

Step	Purpose	Description	Error
Step 8(b): Check ECO operability	An ECO and any of its constituents cannot be operated upon if the user does not have access to the Change Order type of that ECO.	Check if System_Information record has this information. If not, compare Assembly_Type of Change Order Type against value of profile ENG: Engineering Item Change Order Access. If profile value is No, then Assembly_Type must not be Engineering. Set System_Information flag accordingly.	Fatal Error I
Step 8(c): Check operability of parent items and current item (if applicable)	A revised item and any of its components cannot be operated upon if the revised item is implemented or canceled.	Check if System_Information record has this information. If not, find it in the database revised item record, and set System_Information flags accordingly.	Fatal Error III or Fatal Error II (depending on affected entity)
Step 8(d): Check operability of parent items and current item (if applicable)	A revised item and any of its components cannot be operated upon if the user does not have access to the revised item type.	Compare revised item BOM_Item_Type against the revised item access fields in the System_Information record.	Fatal Error III or Fatal Error II (depending on affected entity)

Step	Purpose	Description	Error
Step 9: Value-Id conversions	There are user-friendly value columns that derive certain Id columns. The value columns free up the user from having to enter cryptic Ids, since the Ids can be derived from them.	Derive Ids from user-friendly values	CREATE: Severe Error IV. Other: Standard Error
Step 10: Required Fields checking	Some fields are required for an operation to be performed. Without them, the operation cannot go through. The user must enter values for these fields.	Check that the required field columns are not NULL.	CREATE: Severe Error IV. Other: Standard Error
Step 11: Attribute validation (CREATEs and UPDATEs)	Each of the attributes/fields must be checked individually for validity. Examples of these checks are: range checks, checks against lookups etc.	Check that user-entered attributes are valid. Check each attribute independently of the others	CREATE: Severe Error IV. UPDATE: Standard Error
Step 12: Populate NULL columns(UPDATEs and DELETEs)	The user may send in a record with certain values set to NULL. Values for all such columns are copied over from the OLD record. This feature enables the user to enter minimal information for the operation.	For all NULL columns found in business object record, copy over values from OLD record.	-N/A-

Step	Purpose	Description	Error
Step 13: Default values for NULL attributes (CREATEs)	For CREATEs, there is no OLD record. So the program must default in individual attribute values, independently of each other. This feature enables the user to enter minimal information for the operation to go through.	For all NULL columns found in business object record, try to default in values, either by retrieving them from the database, or by having the program assign values.	Severe Error IV
Step 14: Check conditionally required attributes	Some attributes are required based on certain external factors such as the Transaction_Type value.	Perform checks to confirm all conditionally required attributes are present.	Severe Error IV
Step 15: Entity level defaulting	Certain column values may depend on profile options, other columns in the same table, columns in other tables, etc. Defaulting for these columns happens here.	For all NULL columns in record, try to default in values based on other values. Set all MISSING column values to NULL.	CREATE: Severe Error IV. UPDATE: Standard Error

Step	Purpose	Description	Error
Step 16: Entity level validation	This is where the whole record is checked. The following are checked: Non-updateable columns (UPDATEs): Certain columns must not be changed by the user when updating the record. Cross-attribute checking: The validity of attributes may be checked, based on factors external to it. Business logic: The record must comply with business logic rules.	Perform checks against record in the order specified in the -Purpose- column.	CREATE: Severe Error IV. UPDATE: Standard Error
Step 17: Database writes	Write record to database table.	Perform database write: Insert record for CREATE Overwrite record for UPDATE and CANCEL Remove record for DELETE	-N/A-

Step	Purpose	Description	Error
Step 18(a): Process direct and indirect children	The program will finish processing an entire branch before moving on to a sibling branch. A branch within the business object tree consists of all direct and indirect children.	Pick up the first un-processed child record and Go to Step 5. Continue until all direct children have been processed. Then pick up the first un-processed indirect child record and do the same as above. When no more records are found, Go to Step 20.	-N/A-
Step 18(b): Process siblings	When an entire branch of a record has been processed, the siblings of the current record are processed. The sibling may also contain a branch. So the processing for the sibling will be exactly the same as the current record.	Pick up the first un-processed sibling record and Go to Step 5. Continue through the loop until all siblings have been processed. When no more records are found, Go to Step 21.	-N/A-
Step 18 (c): Process parent record's siblings	Once all the siblings have been processed, the program will move up to the parent (of this entire branch) and process all of its siblings (which will contains branches of their own).	Go up to parent and pick up the first un-processed sibling of the parent. Go to Step 5. Continue through the loop until all siblings have been processed. When there are no more records, Go to Step 4.	-N/A-

Entity Process Flows

Exposed Columns and Un-exposed columns

Each business object record that the user includes in the business object needs user-input for certain columns (required fields, conditionally required fields, user-unique index columns). There are also some other columns that are user-enterable for the import program. These columns are called exposed columns, and are included in the data structures that the user passes the business object records in.

All columns that exist in the Production table, but are not exposed to the user are called un-exposed columns. These columns are not exposed either because they are not user-enterable, or because no user-input is required. In the latter case, the import program saves on processing, since user-input will require validation that the program can avoid by defaulting values into these columns by itself.

Data Entry Rules

- Certain fields are REQUIRED fields. They cannot be derived or defaulted. So the user must enter values for these fields.
- Transaction_Type is always required. It specifies what operation is to be performed on a record. When left empty (NULL), this will cause the record to error out. Also, this field must have values (CREATE, UPDATE, or DELETE). In addition, it can also have the value CANCEL for the Revised Component entity only.
- Each column can take in missing or null values. Defaulting only happens for columns with NULL values, unless the business logic explicitly requires specific defaulting. On the other hand, a column with a missing value is NULL-ed out. The following are the missing values for the different data types:
  - Varchar2 : chr(12)
  - Int : 9.99E125
  - Date : TO_DATE('1','j')

ECO Headers

Exposed Name	Actual Column Name	Type	Comments
ECO_Name	Change_Notice	VARCHAR2(10)	User-Unique Index
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Transaction_Type	-N/A-	VARCHAR2(30)	Required
Change_Type_Code	Change_Order_Type	VARCHAR2(10)	Required for Create
Status_Type	Status_Type	NUMBER	
ECO_Department_Name	Responsible_Organization_Id	VARCHAR2(3)	
Priority_Code	Priority_Code	VARCHAR2(10)	
Approval_List_Name	Approval_List_Name	VARCHAR2(10)	
Approval_Status_Type	Approval_Status_Type	NUMBER	
Reason_Code	Reason_Code	VARCHAR2(10)	
ENG_Implementation_Cost	Estimated_ENG_Cost	NUMBER	
MFG_Implementation_Cost	Estimated_MFG_Cost	NUMBER	
Cancellation_Comments	Cancellation_Comments	VARCHAR2(240)	
Requestor	Requestor_Id	NUMBER	Requestor = Employee Number
Description	Description	VARCHAR2(2000)	
Return_Status	-N/A-	VARCHAR2(1)	
Attribute Category	Attribute Category	VARCHAR2(30)	

Exposed Name	Actual Column Name	Type	Comments
Attribute1	Attribute1	VARCHAR2(150)	
Attribute2	Attribute2	VARCHAR2(150)	
Attribute3	Attribute3	VARCHAR2(150)	
Attribute4	Attribute4	VARCHAR2(150)	
Attribute5	Attribute5	VARCHAR2(150)	
Attribute6	Attribute6	VARCHAR2(150)	
Attribute7	Attribute7	VARCHAR2(150)	
Attribute8	Attribute8	VARCHAR2(150)	
Attribute9	Attribute9	VARCHAR2(150)	
Attribute10	Attribute10	VARCHAR2(150)	
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

ECO Revisions

Columns Exposed to the User

Name	Actual Name	Type	Comments
ECO_Name	Change_Notice	VARCHAR2(10)	User-Unique Index
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Revision	Revision	VARCHAR2(10)	User-Unique Index
New_Revision	Revision	VARCHAR2(10)	Allows updates to Revision
Transaction_Type	-N/A-	VARCHAR2(30)	Required
Comments	Comments	VARCHAR2(240)	
Return_Status	-N/A-	VARCHAR2(30)	
Attribute_Category	Attribute_Category	VARCHAR2(30)	
Attribute1	Attribute1	VARCHAR2(150)	
Attribute2	Attribute2	VARCHAR2(150)	
Attribute3	Attribute3	VARCHAR2(150)	
Attribute4	Attribute4	VARCHAR2(150)	
Attribute5	Attribute5	VARCHAR2(150)	
Attribute6	Attribute6	VARCHAR2(150)	
Attribute7	Attribute7	VARCHAR2(150)	
Attribute8	Attribute8	VARCHAR2(150)	
Attribute9	Attribute9	VARCHAR2(150)	
Attribute10	Attribute10	VARCHAR2(150)	
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	

Name	Actual Name	Type	Comments
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Revised Items

Columns Exposed to the User

Name	Assoc. Column Name	Type	Comments
ECO_Name	Change_Notice	VARCHAR2(10)	User-Unique Index
Organization_Code	Organization_Code	VARCHAR2(3)	User-Unique Index
Revised_Item_Name	Revised_Item_Id	VARCHAR2(81)	User-Unique Index
New_Revised_Item_Revision	New_Item_Revision	VARCHAR2(3)	User-Unique Index
Updated_Revised_Item_Revision	New_Item_Revision	VARCHAR2(3)	Allows updates to New_Revised_Item_Revision
Start_Effective_Date	Scheduled_Date	DATE	User-Unique Index
New_Effective_Date	Scheduled_Date	DATE	Allows updates to Effectivity_Date
Transaction_Type	-N/A-	VARCHAR2(30)	Required
Alternate_BOM_Code	Alternate_BOM_Designator	VARCHAR2(10)	
Status_Type	Status_Type	NUMBER	

Name	Assoc. Column Name	Type	Comments
MRP_Active	MRP_Active	NUMBER	1/yes 2/no
Earliest_Effective_Date	Early_Schedule_Date	DATE	
Use_Up_Item_Name	Use_Up_Item_Id	VARCHAR2(81)	
Use_Up_Plan_Name	Use_Up_Plan_Name	VARCHAR2(10)	
Disposition_Type	Disposition_Type	NUMBER	
Update_WIP	Update_WIP	NUMBER	1/yes 2/no
Cancel_Comments	Cancel_Comments	VARCHAR2(240)	
Descriptive_Text	Change_Description	VARCHAR2(240)	
Attribute Category	Attribute Category	VARCHAR2(30)	
Attribute1	Attribute1	VARCHAR2(150)	
Attribute2	Attribute2	VARCHAR2(150)	
Attribute3	Attribute3	VARCHAR2(150)	
Attribute4	Attribute4	VARCHAR2(150)	
Attribute5	Attribute5	VARCHAR2(150)	
Attribute6	Attribute6	VARCHAR2(150)	
Attribute7	Attribute7	VARCHAR2(150)	
Attribute8	Attribute8	VARCHAR2(150)	
Attribute9	Attribute9	VARCHAR2(150)	
Attribute10	Attribute10	VARCHAR2(150)	

Name	Assoc. Column Name	Type	Comments
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Note: The New_Revised_Item_Revision is not updateable for revised items since it is part of the unique key which uniquely identifies a record. So, updates to it have to be made by entering the new revision into Updated_Revised_Item_Revision. After the record is retrieved using the unique key, its revision is overwritten by the new value.

- Just like New_Revised_Item_Revision, Start_Effective_Date is a unique index column. So changes to it have to be made by entering the new value into New_Effective_Date.
- The user can reschedule a revised item by entering the new Effective Date in New_Effective_Date. All Effective Date changes will be recorded in the history table ENG_Current_Start_Effective_Dates, along with requestor_id and comments.
- Revised item revisions go into MTL_ITEM_REVISIONS. When an item is first defined through the Define Item form, a starting revision record is written out to this table. Any subsequent revisions are added to or updated in MTL_ITEM_REVISIONS.

Revised Components

Columns Exposed to the User

Name	Associated Column Name	Type	Comments
ECO_Name	ECO_Name	VARCHAR2(10)	User-Unique Index
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Revised_Item_Name	Revised_Item_Id	VARCHAR2(81)	User-Unique Index
New_Revised_Item_Revision	New_Item_Revision	VARCHAR2(3)	User-Unique Index
Start_Effective_Date	Effectivity_Date	DATE	User-Unique Index
Operation_Sequence_Number	Operation_Seq_Num	NUMBER	User-Unique Index
Component_Item_Name	Component_Item_Id	VARCHAR2(81)	User-Unique Index
Alternate_BOM_Code	Alternate_BOM_Designator	VARCHAR2(10)	User-Unique Index
Transaction_Type	-N/A-	VARCHAR2(30)	Required
ACD_Type	ACD_Type	NUMBER	Required. 1/add 2/change 3/disable
Old_Effectivity_Date	Effectivity_Date	DATE	
Old_Operation_Sequence_Number	Operation_Seq_Num	NUMBER	
New_Operation_Sequence_Number	Operation_Seq_Num	NUMBER	Allows updates to Operation_Seq_Num
Item_Sequence_Number	Item_Num	NUMBER	
Quantity_Per_Assembly	Component_Quantity	NUMBER	
Planning_Percent	Planning_Factor	NUMBER	

Name	Associated Column Name	Type	Comments
Projected_Yield	Component_Yield_Factor	NUMBER	
Include_In_Cost_Rollup	Include_In_Cost_Rollup	NUMBER	1/yes 2/no
WIP_Supply_Type	WIP_Supply_Type	NUMBER	
Supply_Subinventory	Supply_Subinventory	VARCHAR2(10)	
Locator_Name	Locator_Id	VARCHAR2(81)	
SO_Basis	SO_Basis	NUMBER	1/yes 2/no
Optional	Optional	NUMBER	1/yes 2/no
Mutually_Exclusive	Mutually_Exclusive_Options	NUMBER	1/yes 2/no
Check_ATP	Check_ATP	NUMBER	1/yes 2/no
Minimum_Allowed_Quantity	Low_Quantity	NUMBER	
Maximum_Allowed_Quantity	High_Quantity	NUMBER	
Shipping_Allowed	Shipping_Allowed	NUMBER	1/yes 2/no
Required_To_Ship	Required_To_Ship	NUMBER	1/yes 2/no
Required_For_Revenue	Required_For_Revenue	NUMBER	1/yes 2/no
Include_On_Ship_Docs	Include_On_Ship_Docs	NUMBER	1/yes 2/no
Comments	Component_Remarks	VARCHAR2(240)	
Quantity_Related	Quantity_Related	NUMBER	1/yes 2/no

Name	Associated Column Name	Type	Comments
Return_Status	-N/A-	VARCHAR2(1)	
Attribute Category	Attribute Category	VARCHAR2(30)	
Attribute1	Attribute1	VARCHAR2(150)	
Attribute2	Attribute2	VARCHAR2(150)	
Attribute3	Attribute3	VARCHAR2(150)	
Attribute4	Attribute4	VARCHAR2(150)	
Attribute5	Attribute5	VARCHAR2(150)	
Attribute6	Attribute6	VARCHAR2(150)	
Attribute7	Attribute7	VARCHAR2(150)	
Attribute8	Attribute8	VARCHAR2(150)	
Attribute9	Attribute9	VARCHAR2(150)	
Attribute10	Attribute10	VARCHAR2(150)	
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Note: The ECO Header unique key is part of the revised item unique key. So confirming the revised item confirms the revised item-ECO Header link, and, consequently, the Header's link to all other entity records.

Start_Effectivity_Date is not updateable. So Effectivity Date changes have to made using the parent revised item's New_Effective_Date column. Please note however, that revised item Effectivity Date changes will be propagated to all un-implemented child revised components.

Users can cancel Revised Components by entering a Transaction_Type value of CANCEL.

Item locators cannot be created dynamically. There are no APIs to validate flexfields.

Adding a component to a bill-less revised item causes a new primary bill to be created.

For ACD_Type = ('Change' or 'Disable'), Old_Operation_Sequence_Number identifies the implemented record being changed or disabled. Operation_Sequence_Number identifies the current un-implemented record. New_Operation_Sequence_Number allows user-updates to Operation_Sequence_Number. For acd_type = 'Add', Old_Operation_Sequence_Number will equal Operation_Sequence_Number, and New_Operation_Sequence_Number will be null.

Reference Designators

Columns Exposed to the User

Name	Associated Column Name	Type	Comments
ECO_Name	ECO_Name	VARCHAR2(10)	User-Unique Index
Organization_Code	Organization_Code	VARCHAR2(3)	User-Unique Index
Revised_Item_Name	Revised_Item_Id	VARCHAR2(81)	User-Unique Index
Start_Effective_Date	Effectivity_Date	VARCHAR2(3)	User-Unique Index
New_Revised_Item_Revision	New_Item_Revision	DATE	User-Unique Index
Operation_Sequence_Number	Operation_Seq_Num	NUMBER	User-Unique Index
Alternate_BOM_Code	Alternate_BOM_Designator	VARCHAR2(10)	User-Unique Index
Component_Item_Name	Component_Item_Id	VARCHAR2(81)	User-Unique Index

Name	Associated Column Name	Type	Comments
Reference_Designator_Name	Component_Reference_Designator	VARCHAR2(15)	User-Unique Index
ACD_Type	ACD_Type	NUMBER	User-Unique Index
Transaction_Type	-N/A-	VARCHAR2(30)	Required
Ref_Designator_Comment	Ref_Designator_Comment	VARCHAR2(240)	
Return_Status	-N/A-	VARCHAR2(1)	
Attribute Category	Attribute Category	VARCHAR2(30)	
Attribute1	Attribute1	VARCHAR2(150)	
Attribute2	Attribute2	VARCHAR2(150)	
Attribute3	Attribute3	VARCHAR2(150)	
Attribute4	Attribute4	VARCHAR2(150)	
Attribute5	Attribute5	VARCHAR2(150)	
Attribute6	Attribute6	VARCHAR2(150)	
Attribute7	Attribute7	VARCHAR2(150)	
Attribute8	Attribute8	VARCHAR2(150)	
Attribute9	Attribute9	VARCHAR2(150)	
Attribute10	Attribute10	VARCHAR2(150)	
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	
Attribute13	Attribute13	VARCHAR2(150)	

Name	Associated Column Name	Type	Comments
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Substitute Components

Columns Exposed to the User

Name	Name	Type	Comments
ECO_Name	Change_Notice	VARCHAR2(10)	User-Unique Index
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Revised_Item_Name	Revised_Item_Id	VARCHAR2(81)	User-Unique Index
Start_Effective_Date	Effectivity_Date	VARCHAR2(3)	User-Unique Index
New_Revised_Item_Revision	New_Item_Revision	DATE	User-Unique Index
Operation_Sequence_Number	Operation_Seq_Num	NUMBER	User-Unique Index
Alternate_BOM_Code	Alternate_BOM_Designator	VARCHAR2(10)	User-Unique Index
Component_Item_Name	Component_Item_Id	VARCHAR2(81)	User-Unique Index
Substitute_Component_Name	Substitute_Component_Id	VARCHAR2(81)	User-Unique Index
ACD_Type	ACD_Type	NUMBER	User-Unique Index
Transaction_Type	-N/A	VARCHAR2(30)	Required

Name	Name	Type	Comments
Substitute_Item_Quantity	Substitute_Item_Quantity	NUMBER	
Return_Status	-N/A-	VARCHAR2(1)	
Attribute Category	Attribute Category	VARCHAR2(30)	
Attribute1	Attribute1	VARCHAR2(150)	
Attribute2	Attribute2	VARCHAR2(150)	
Attribute3	Attribute3	VARCHAR2(150)	
Attribute4	Attribute4	VARCHAR2(150)	
Attribute5	Attribute5	VARCHAR2(150)	
Attribute6	Attribute6	VARCHAR2(150)	
Attribute7	Attribute7	VARCHAR2(150)	
Attribute8	Attribute8	VARCHAR2(150)	
Attribute9	Attribute9	VARCHAR2(150)	
Attribute10	Attribute10	VARCHAR2(150)	
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

New API Packages

The ECO Business object interface program will contain new Business Object API

packages. Please see the Business Object Framework section for the flow of control in these packages.

Main Packages

- ENG_Eco_PUB : This is the public PL/SQL package that is to be programmatically called to launch the import. This package ensures that all records belong to the same business object before it lets the business object through to the Private package.
- ENG_Eco_PVT : This is the private PL/SQL package. It performs all the business logic by calling the Entity and Shared packages as and when required.

Shared Packages

- ENG_Globals : This Global package contains all global structure and variable definitions, as well as commonly used procedures and functions.
- ENG_Validate : This is a shared validation package. This package contains all attribute-level validation logic.
- ENG_Value_to_ID : This is the shared Value-Id conversion package. This package contains all Value-Id conversions.
- Error_Handler : This is the shared error handling package. This package is responsible for logging error messages and erroring out records.

Entity Packages

- ENG_Validate_Eco: This package contains Attribute and Entity level validation for the ECO Header.
- ENG_Default_Eco: This package contains default attribute procedures for the ECO Header.
- ENG_Eco_Util: This package contains entity utility functions and procedures for the ECO Header.
- ENG_Validate_EcoRevision: This package contains Attribute and Entity level validation for the ECO Revision.
- ENG_Default_EcoRevision: This package contains default attribute procedures for the ECO Revision.
- ENG_EcoRevision_UTIL: This package contains entity utility functions and procedures for the ECO Revision.

- ENG_Validate_RevisedItem: This package contains Attribute and Entity level validation for the Revised Item.
- ENG_Default_RevisedItem: This package contains default attribute procedures for the Revised Item.
- ENG_RevisedItem_UTIL: This package contains entity utility functions and procedures for the Revised Item.
- BOM_Validate_RevisedComp: This package contains Attribute and Entity level validation for the Revised Component.
- BOM_Default_RevisedComp: This package contains default attribute procedures for the Revised Component.
- BOM_RevisedComp_UTIL: This package contains entity utility functions and procedures for the RevisedComponent.
- BOM_Validate_ReferenceDesig: This package contains Attribute and Entity level validation for the Reference Designator.
- BOM_Default_ReferenceDesig: This package contains default attribute procedures for the Reference Designator.
- BOM_ReferenceDesig_UTIL: This package contains entity utility functions and procedures for the Reference Designator.
- BOM_Validate_SubstituteComp: This package contains Attribute and Entity level validation for the Substitute Component.
- BOM_Default_SubstituteComp: This package contains default attribute procedures for the Substitute Component.
- BOM_SubstituteComp_UTIL: This package contains entity utility functions and procedures for the Substitute Component.

Commits and Rollbacks

None of the ECO business object APIs issue commits or rollbacks. It is the responsibility of the calling code to issue them. This ensures that parts of the transaction are not left in the database. If an error occurs, the whole transaction is rolled back. Therefore, API work is either all completed or none of the work is done.

The APIs pass back to the caller, the state of the business object, whether there has been an error, or it has been successfully processed. The caller could use this to perform further activities by building on top of the APIs. This gives callers the flexibility to issue commits and rollbacks where they decide.

Launching the Import

The Three Step Rule:

To use the business object APIs effectively, the user must follow the three step rule:

1. Initialize the system information.
2. Call the Public API.
3. Review all relevant information after the Public API has completed.

Step1: Initialize the system information

Each database table that the program writes to requires system information, such as who is trying to update the current record. The user must provide this information to the import program by initializing certain variables. The program will retrieve system information from these variables, during operation. To initialize the variables, the user must call the following procedure:

FND_GLOBAL.apps_initialize

(user_id IN NUMBER

, resp_id IN NUMBER

, resp_appl_id IN NUMBER

, security_group_id IN NUMBER DEFAULT 0

)

Pointers:

1. This procedure initializes the global security context for each database session.
2. This initialization should be done when the session is established outside of a normal forms or concurrent program connection.
3. user_id is the FND User Id of the person launching this program.
4. resp id is the FND Responsibility Id the person is using.
5. resp_appl_id is the Application Responsibility Id.
6. security_group_id is the FND Security Group Id.

Step2: Call the Public API

The Public API is the user's interface to the Import program. The user must call it programmatically, while sending in one business object at a time. The Public API returns the processed business object, the business object status, and a count of all associated error and warning messages.

The procedure to call is Process_Eco, and the following is its specification :

```
PROCEDURE Process_Eco
( p_api_version_number IN NUMBER
, p_init_msg_list IN VARCHAR2 := FND_API.G_FALSE
, x_return_status OUT VARCHAR2
, x_msg_count OUT NUMBER
, x_msg_data OUT VARCHAR2
, p_ECO_rec IN Eco_Rec_Type := G_MISS_ECO_REC
, p_eco_revision_tbl IN Eco_Revision_Tbl_Type := G_MISS_ECO_REVISION_TBL
, p_revised_item_tbl IN Revised_Item_Tbl_Type := G_MISS_REVISED_ITEM_TBL
, p_rev_component_tbl IN Rev_Component_Tbl_Type :=
G_MISS_REV_COMPONENT_TBL
, p_ref_designator_tbl IN Ref_Designator_Tbl_Type :=
G_MISS_REF_DESIGNATOR_TBL
, p_sub_component_tbl IN Sub_Component_Tbl_Type :=
G_MISS_SUB_COMPONENT_TBL
, x_ECO_rec OUT Eco_Rec_Type
, x_eco_revision_tbl OUT Eco_Revision_Tbl_Type
, x_revised_item_tbl OUT Revised_Item_Tbl_Type
, x_rev_component_tbl OUT Rev_Component_Tbl_Type
, x_ref_designator_tbl OUT Ref_Designator_Tbl_Type
, x_sub_component_tbl OUT Sub_Component_Tbl_Type
)
```

As is obvious from the above specification, all IN parameters begin with p_. All OUT parameters begin with x_. The following is a description of these parameters :

- p_api_version_number : This parameter is required. It is used by the API to compare the version numbers of incoming calls to its current version number, and return an unexpected error if they are incompatible. See section 4.1 of the Business Object Coding Standards document for a detailed description of this parameter.

- p_init_msg_list : This parameter, if set to TRUE, allows callers to request that the API do the initialization of the message list on their behalf. On the other hand, the caller may set this to FALSE (or accept the default value) in order to do the initialization itself by calling FND_MSG_PUB.Initialize.

- p_eco_rec, p_eco_revision_tbl, p_revised_item_tbl, p_rev_component_tbl, p_ref_designator_tbl, p_sub_component_tbl : This is the set of data structures that

represents the incoming business object. p_eco_rec is a record that holds the ECO header for the ECO. All the other data structures are PL/SQL tables of records that hold records for each of the other entities. All these data structures directly correspond to the entities shown in the ECO entity diagram.

Please note that any of these data structures may be sent in empty (set to NULL) to indicate that there are no instances of that entity in the business object being sent in.

- x_eco_rec, x_eco_revision_tbl, x_revised_item_tbl, x_rev_component_tbl, x_ref_designator_tbl, x_sub_component_tbl : This is the set of data structures that represents the outgoing business object. These records essentially constitute the whole business object as it was sent in, except now it has all the changes that the import program made to it through all the steps in the Process Flow. These records can be committed, rolled back, or subjected to further processing by the caller. x_eco_rec is a record that holds the ECO header for the ECO. All the other data structures are PL/SQL tables of records that hold records for each of the other entities. All these data structures directly correspond to the entities shown in the ECO entity diagram.

- x_return_status : This is a flag that indicates the state of the whole business object after the import. If there has been an error in a record, this status will indicate the fact that there has been an error in the business object. Similarly, if the business object import has been successful, this flag will carry a status to indicate that. The caller may look up this flag to choose an appropriate course of action (commit, rollback, or further processing by the caller). The following is a list of all the possible business object states:

- x_msg_count : This holds the number of messages in the API message stack after the import. This parameter returns a 0 when there are no messages to return.

- x_msg_data : This returns a single message when the message count is 1. The purpose of this parameter is to save the caller the extra effort of retrieving a lone message from the message list. This parameter returns NULL when there is more than one message.

As mentioned above, the Public API must be called programmatically. The caller here may be a form, a shell, or some other API which serves to extend the functionality of the import program. Please see the Sample Shell section in the Appendix for a complete listing of the shell that was written to test the import program. This shell illustrates the correct usage of the import program.

> **Note:** A record must have an error status of NULL for it to be processed. If it has any other value, it will not be picked up for processing. The user must remember to NULL out this field when sending in a record.

Step 3: Review all relevant information after the Public API has completed

The user must look up:

- all error status that the Public API returns, including, the overall business object error status, as well as the individual record statuses.
- all record attributes to see what changes occurred after applying business logic to these records.
- all error and warning messages in the API message list.

The user can access the API message list using the following procedures and functions in the Error_Handler package:

1. Initializing the message list: The following procedure clears the message list and initializes all associated variables

   PROCEDURE Initialize;

2. Go to the start of the list: The following procedure reset the message index to the start of the list so the user can start reading from the start of the list

   PROCEDURE Reset;

3. Retrieving the entire message list: The following procedure will return the entire message list to the user

   PROCEDURE Get_Message_List

   (x_message_list OUT Eng_Eco_Pub.Error_Tbl_Type);

4. Retrieving messages by entity: One implementation of procedure Get_Entity_Message will return all messages pertaining to a particular entity (p_entity_id), denoted by the symbols ECO (ECO Header), RI (Revised Item), RC (Revised Component), SC (Substitute Component), RD (Reference Designator).

   PROCEDURE Get_Entity_Message

   (p_entity_id IN VARCHAR2

   , x_message_list OUT Eng_Eco_Pub.Error_Tbl_Type

   );

5. Retrieving a specific message: Another implementation of procedure Get_Entity_Message will return the message pertaining to a particular entity (p_entity_id), at a specific array index in that entity table. The entity is denoted by the symbols ECO (ECO Header), RI (Revised Item), RC (Revised Component), SC (Substitute Component), RD (Reference Designator). The entity index (p_entity_index) is the index in the entity array.

   PROCEDURE Get_Entity_Message

   (p_entity_id IN VARCHAR2

, p_entity_index IN NUMBER

, x_message_text OUT VARCHAR2

);

6. Retrieving the current message: Procedure Get_Message will return the message at the current message index and will advance the pointer to the next index. If the user tries to retrieve beyond the size of the message list, then the message index will be reset to the beginning of the list.

   PROCEDURE Get_Message

   (x_message_text OUT VARCHAR2

   , x_entity_index OUT NUMBER

   , x_entity_id OUT VARCHAR2

   );

7. Deleting a specific message: Procedure Delete_Message enables the user to delete a specific message at a specified entity index (p_entity_index) within the PL/SQL table of a specified entity (p_entity_id). The entity is denoted by the symbols ECO (ECO Header), RI (Revised Item), RC (Revised Component), SC (Substitute Component), RD (Reference Designator). The entity index (p_entity_index) is the index in the entity array.

   PROCEDURE Delete_Message

   (p_entity_id IN VARCHAR2

   , p_entity_index IN NUMBER

   );

8. Deleting all messages for a certain entity: Another implementation of procedure Delete_Message lets the user delete all messages for a particular entity (p_entity_id). The entity is denoted by the symbols ECO (ECO Header), RI (Revised Item), RC (Revised Component), SC (Substitute Component), RD (Reference Designator).

   PROCEDURE Delete_Message

   (p_entity_id IN VARCHAR2);

9. Get a count of all messages: The following functions returns the total number of messages currently in the message list

   FUNCTION Get_Message_Count RETURN NUMBER;

10. Dumping the message list: The following message generates a dump of the message list using dbms_output

    PROCEDURE Dump_Message_List;

Package Interaction

The Public Package - ENG_ECO_PUB

This package is like a gatekeeper, letting only one business object through at a time. This essentially means that all records in the business object must belong to the business object. The business object here is the ECO, and incoming records together make up an instance of the business object. So, all records in an ECO business object instance must reference the same ECO.

Main Procedure: Process_ECO

1. Set business object status to 'S'.

2. Check that all records in the business object belong to the same Bill, i.e., all records must have the same Assembly Item Name and Organization_Code combination

Description	Cause of Failure	Error	Message
If there is an ECO Header in the business object, check that all records have the same ECO_name and Organization_Code values as the Header. If the business object does not have an ECO Header record, check that all records have the same ECO_Name and Organization_Code combination as the first highest level entity record picked up.	Any records have mismatched ECO_Name and Organization_Code values	Severe Error I	ENG_MUST_BE_IN_SAME_ECO:All records in a business object must belong to the same ECO. That is, they must all have the same ECO Name and organization. Please check your records for this.

3. Derive Organization_Id from Organization_Code and copy this value into all business object records.

Column	Description	Error	Message
Organization_Id	Derive using Organization_Code from table MTL_PARAMTERS	Severe Error I	ENG_ORG_INVALID:The Organization <org_id> you entered is invalid.

Unexpected Error Other Message: ENG_UNEXP_ORG_INVALID: This record was not processed since an unexpected error while performing a value to id conversion for organization code.

4. Pass business object into Private API if the business object status is still 'S'. Also pass the Assembly Item Name and Organization_Id to Private API, to identify this business object instance.

5. Accept processed business object and return status from Private API after the import, and pass it back to the calling program.

The Private Package - ENG_BO_PVT

This package is called by the Public package. It carries out all the business object checks listed in the Process Flow, while making any necessary changes to it, and also performs production tables inserts, updates and deletes. It then passes the business object and the business object import status back to the Public API.

Main Procedure: Process_ECO

1. Initialize User_Id, Login_Id, Prog_AppId, Prog_Id in System_Information record.

Column	Description
User_Id	From environment
Login_Id	From environment
Prog_AppId	From environment
Prog_Id	From environment

2. Initialize Assembly_Item_Name and Org_Id in System_Information record from values passed by Public API.

3. If an BOM Header was passed in, call ECO_Header

4. If BOM Revisions records exist, call ECO_Rev
5. Call Rev_Comps to process immediate-parentless components
6. Call Ref_Desgs to process immediate-parentless reference designators
7. Call Sub_Comps to process immediate-parentless substitute components
8. Return import status and processed business object to Public API

The sections below list the steps performed within each of the entity procedures: ECO_Header, ECO_Rev, Rev_Comps, Ref_Desgs, Sub_Comps. Each of these entity procedures performs all the entity process flow steps listed by entity under the Entity Process Details section. They also call other entity procedures to process child records.

The entity procedure descriptions below also point out how the Private API reacts to errors in entity process flow steps.

Sample Launch Package

The following is the PL/SQL program that was coded to test the APIs. its purpose here is to illustrate how the user might call the Public API.

/***
****

This package calls the Public API. It uses interface table ENG_ENG_CHANGES_INTERFACE, ENG_ECO_REVISIONS_INTERFACE, ENG_REVISED_ITEMS_INTERFACE, BOM_INVENTORY_COMPS_INTERFACE, BM_REF_DESGS_INTERFACE and BOM_SUB_COMPS_INTERFACE. The way it works is that it assumes that user has loaded these tables with business object records, and then runs this program. Each record will have a TEST TAG which identifies the whole business object uniquely to the user.

When the user runs this program, he/she specifies a TEST TAG (p_test_tag). This program picks up all records that carry this test tag value as one business object, and then tries to import it into the production tables.

****
*******************************/

```
CREATE OR REPLACE

PROCEDURE Public_API_UT

( p_test_tag IN VARCHAR2 )

IS

l_eco_rec Eng_Eco_Pub.Eco_Rec_Type;

l_eco_revision_tbl Eng_Eco_Pub.Eco_Revision_Tbl_Type;
```

```
l_revised_item_tbl Eng_Eco_Pub.Revised_Item_Tbl_Type;
l_rev_component_tbl Eng_Eco_Pub.Rev_Component_Tbl_Type;
l_sub_component_tbl Eng_Eco_Pub.Sub_Component_Tbl_Type;
l_ref_designator_tbl Eng_Eco_Pub.Ref_Designator_Tbl_Type;
l_return_status VARCHAR2(1);
l_msg_count NUMBER;
l_msg_data VARCHAR2(2000);
l_Error_Table Eng_Eco_Pub.Error_Tbl_Type;
l_Message_text VARCHAR2(2000);
CURSOR c_eco_rec IS
SELECT *
FROM eng_eng_changes_interface
WHERE eng_changes_ifce_key like p_test_tag;
CURSOR c_eco_rev IS
SELECT *
FROM eng_eco_revisions_interface
WHERE eng_eco_revisions_ifce_key like p_test_tag;
CURSOR c_rev_items IS
SELECT *
FROM eng_revised_items_interface
WHERE eng_revised_items_ifce_key like p_test_tag;
CURSOR c_rev_comps IS
SELECT *
FROM bom_inventory_comps_interface
WHERE bom_inventory_comps_ifce_key like p_test_tag;
CURSOR c_sub_comps IS
SELECT *
FROM bom_sub_comps_interface
WHERE bom_sub_comps_ifce_key like p_test_tag;
CURSOR c_ref_desgs IS
SELECT *
FROM bom_ref_desgs_interface
```

```
WHERE bom_ref_desgs_ifce_key like p_test_tag;
i number;
BEGIN
-- Query all the records and call the Private API.
FOR eco_rec IN c_eco_rec
LOOP
l_eco_rec.eco_name := eco_rec.change_notice;
l_eco_rec.organization_code := eco_rec.organization_code;
l_eco_rec.change_type_code := eco_rec.change_order_type;
l_eco_rec.status_type := eco_rec.status_type;
l_eco_rec.eco_department_name := eco_rec.responsible_org_code;
l_eco_rec.priority_code := eco_rec.priority_code;
l_eco_rec.approval_list_name := eco_rec.approval_list_name;
l_eco_rec.approval_status_type := eco_rec.approval_status_type;
l_eco_rec.reason_code := eco_rec.reason_code;
l_eco_rec.eng_implementation_cost := eco_rec.estimated_eng_cost;
l_eco_rec.mfg_implementation_cost := eco_rec.estimated_mfg_cost;
l_eco_rec.cancellation_comments:=eco_rec.cancellation_comments;
l_eco_rec.requestor := eco_rec.requestor;
l_eco_rec.description := eco_rec.description;
l_eco_rec.transaction_type := eco_rec.transaction_type;
END LOOP;
-- Fetch ECO Revisions
i := 1;
FOR rev IN c_eco_rev
LOOP
l_eco_revision_tbl(i).eco_name := rev.change_notice;
l_eco_revision_tbl(i).organization_code:= rev.organization_code;
l_eco_revision_tbl(i).revision := rev.revision;
l_eco_revision_tbl(i).new_revision := rev.new_revision;
l_eco_revision_tbl(i).comments := rev.comments;
l_eco_revision_tbl(i).transaction_type := rev.transaction_type;
```

```
i := i + 1;
END LOOP;
-- Fetch revised items
i := 1;
FOR ri IN c_rev_items
LOOP
l_revised_item_tbl(i).eco_name := ri.change_notice;
l_revised_item_tbl(i).organization_code := ri.organization_code;
l_revised_item_tbl(i).revised_item_name :=
ri.revised_item_number;
IF ri.new_item_revision = FND_API.G_MISS_CHAR
THEN
l_revised_item_tbl(i).new_revised_item_revision := NULL;
ELSE
l_revised_item_tbl(i).new_revised_item_revision :=
ri.new_item_revision;
END IF;
l_revised_item_tbl(i).start_effective_date :=
ri.scheduled_date;
l_revised_item_tbl(i).alternate_bom_code :=
ri.alternate_bom_designator;
l_revised_item_tbl(i).status_type := ri.status_type;
l_revised_item_tbl(i).mrp_active := ri.mrp_active;
l_revised_item_tbl(i).earliest_effective_date :=
ri.early_schedule_date;
l_revised_item_tbl(i).use_up_item_name := ri.use_up_item_number;
l_revised_item_tbl(i).use_up_plan_name := ri.use_up_plan_name;
l_revised_item_tbl(i).disposition_type := ri.disposition_type;
l_revised_item_tbl(i).update_wip := ri.update_wip;
l_revised_item_tbl(i).cancel_comments := ri.cancel_comments;
l_revised_item_tbl(i).change_description := ri.descriptive_text;
l_revised_item_tbl(i).transaction_type := ri.transaction_type;
```

```
i := i + 1;
END LOOP;
-- Fetch revised components
i := 1;
FOR rc IN c_rev_comps
LOOP
l_rev_component_tbl(i).eco_name := rc.change_notice;
l_rev_component_tbl(i).organization_code:= rc.organization_code;
l_rev_component_tbl(i).revised_item_name :=
rc.assembly_item_number;
l_rev_component_tbl(i).new_revised_item_revision := NULL;
l_rev_component_tbl(i).start_effective_date :=
rc.effectivity_date;
l_rev_component_tbl(i).disable_date := rc.disable_date;
l_rev_component_tbl(i).operation_sequence_number :=
rc.operation_seq_num;
l_rev_component_tbl(i).component_item_name :=
rc.component_item_number;
l_rev_component_tbl(i).alternate_bom_code :=
rc.alternate_bom_designator;
l_rev_component_tbl(i).acd_type := rc.acd_type;
l_rev_component_tbl(i).old_effectivity_date :=
rc.old_effectivity_date;
l_rev_component_tbl(i).old_operation_sequence_number :=
rc.old_operation_seq_num;
l_rev_component_tbl(i).item_sequence_number := rc.item_num;
l_rev_component_tbl(i).quantity_per_assembly :=
rc.component_quantity;
l_rev_component_tbl(i).planning_percent := rc.planning_factor;
l_rev_component_tbl(i).projected_yield :=
rc.component_yield_factor;
l_rev_component_tbl(i).include_in_cost_rollup :=
```

```
rc.include_in_cost_rollup;
l_rev_component_tbl(i).wip_supply_type := rc.wip_supply_type;
l_rev_component_tbl(i).so_basis := rc.so_basis;
l_rev_component_tbl(i).optional := rc.optional;
l_rev_component_tbl(i).mutually_exclusive :=
rc.mutually_exclusive_options;
l_rev_component_tbl(i).check_atp := rc.check_atp;
l_rev_component_tbl(i).shipping_allowed :=
rc.shipping_allowed;
l_rev_component_tbl(i).required_to_ship := rc.required_to_ship;
l_rev_component_tbl(i).required_for_revenue :=
rc.required_for_revenue;
l_rev_component_tbl(i).include_on_ship_docs :=
rc.include_on_ship_docs;
l_rev_component_tbl(i).quantity_related := rc.quantity_related;
l_rev_component_tbl(i).supply_subinventory :=
rc.supply_subinventory;
l_rev_component_tbl(i).location_name := rc.location_name;
l_rev_component_tbl(i).minimum_allowed_quantity :=
rc.low_quantity;
l_rev_component_tbl(i).maximum_allowed_quantity :=
rc.high_quantity;
l_rev_component_tbl(i).component_remarks :=
rc.component_remarks;
l_rev_component_tbl(i).transaction_type :=
rc.transaction_type;
i := i + 1;
END LOOP;
-- Fetch substitute component records
i := 1;
FOR sc IN c_sub_comps
LOOP
```

```
l_sub_component_tbl(i).eco_name := sc.change_notice;
l_sub_component_tbl(i).organization_code:= sc.organization_code;
l_sub_component_tbl(i).revised_item_name :=
sc.assembly_item_number;
l_sub_component_tbl(i).start_effective_date :=
sc.effectivity_date;
l_sub_component_tbl(i).new_revised_item_revision := NULL;
l_sub_component_tbl(i).component_item_name :=
sc.component_item_number;
l_sub_component_tbl(i).alternate_bom_code :=
sc.alternate_bom_designator;
l_sub_component_tbl(i).substitute_component_name :=
sc.substitute_comp_number;
l_sub_component_tbl(i).acd_type := sc.acd_type;
l_sub_component_tbl(i).operation_sequence_number :=
sc.operation_seq_num;
l_sub_component_tbl(i).substitute_item_quantity :=
sc.substitute_item_quantity;
l_sub_component_tbl(i).transaction_type := sc.transaction_type;
i := i + 1;
END LOOP;
-- Fetch reference designators
i := 1;
FOR rd IN c_ref_desgs
LOOP
l_ref_designator_tbl(i).eco_name := rd.change_notice;
l_ref_designator_tbl(i).organization_code :=
rd.organization_code;
l_ref_designator_tbl(i).revised_item_name :=
rd.assembly_item_number;
l_ref_designator_tbl(i).start_effective_date :=
rd.effectivity_date;
```

```
l_ref_designator_tbl(i).new_revised_item_revision := null;
l_ref_designator_tbl(i).operation_sequence_number :=
rd.operation_seq_num;
l_ref_designator_tbl(i).component_item_name :=
rd.component_item_number;
l_ref_designator_tbl(i).alternate_bom_code :=
rd.alternate_bom_designator;
l_ref_designator_tbl(i).reference_designator_name :=
rd.component_reference_designator;
l_ref_designator_tbl(i).acd_type := rd.acd_type;
l_ref_designator_tbl(i).ref_designator_comment :=
rd.ref_designator_comment;
l_ref_designator_tbl(i).new_reference_designator :=
rd.new_designator;
l_ref_designator_tbl(i).transaction_type :=
rd.transaction_type;
END LOOP;
Eng_Globals.G_WHO_REC.org_id := 207;
Eng_Globals.G_WHO_REC.user_id := 2462;
Eng_Globals.G_WHO_REC.login_id := 2462;
Eng_Globals.G_WHO_REC.prog_appid := 703;
Eng_Globals.G_WHO_REC.prog_id:= NULL;
Eng_Globals.G_WHO_REC.req_id := NULL;
ENG_GLOBALS.system_information.org_id := 207;
fnd_global.apps_initialize
( user_id => Eng_Globals.G_WHO_REC.user_id
, resp_id => 20567
, resp_appl_id => Eng_Globals.G_WHO_REC.prog_appid
);
-- Call the private API
Eng_Eco_PUB.Process_Eco
( p_api_version_number => 1.0
```

```
, x_return_status => l_return_status
, x_msg_count => l_msg_count
, p_eco_rec => l_eco_rec
, p_eco_revision_tbl => l_eco_revision_tbl
, p_revised_item_tbl => l_revised_item_tbl
, p_rev_component_tbl => l_rev_component_tbl
, p_sub_component_tbl => l_sub_component_tbl
, p_ref_designator_tbl => l_ref_designator_tbl
, x_eco_rec => l_eco_rec
, x_eco_revision_tbl => l_eco_revision_tbl
, x_revised_item_tbl => l_revised_item_tbl
, x_rev_component_tbl => l_rev_component_tbl
, x_sub_component_tbl => l_sub_component_tbl
, x_ref_designator_tbl => l_ref_designator_tbl
);
--
-- On return from the PUB API
-- Perform all the error handler operations to verify that the
-- error or warning are displayed and all the error table interface
-- function provided to the user work correctly;
--
Error_Handler.Get_Message_List( x_message_list => l_error_table);
FOR i IN 1..l_error_table.COUNT
LOOP
dbms_output.put_line('Entity Id: '||l_error_table(i).entity_id);
dbms_output.put_line('Index: '||l_error_table(i).entity_index);
dbms_output.put_line('Mesg: '||l_error_table(i).message_text);
dbms_output.put_line('----------------------------------------');
END LOOP;
dbms_output.put_line('Total Messages: ' || to_char(i));
l_msg_count := Error_Handler.Get_Message_Count;
dbms_output.put_line('Message Count Function: '||to_char(l_msg_count));
```

```
Error_Handler.Dump_Message_List;
Error_Handler.Get_Entity_Message
( p_entity_id => 'ECO'
, x_message_list => l_error_table
);
Error_Handler.Get_Entity_Message
( p_entity_id => 'REV'
, x_message_list => l_error_table
);
Error_Handler.Get_Entity_Message
( p_entity_id => 'RI'
, x_message_list => l_error_table
);
Error_Handler.Get_Entity_Message
( p_entity_id => 'RC'
, x_message_list => l_error_table
);
Error_Handler.Get_Entity_Message
( p_entity_id => 'SC'
, x_message_list => l_error_table
);
Error_Handler.Get_Entity_Message
( p_entity_id => 'RD'
, x_message_list => l_error_table
);
END Public_API_UT;
```

Import Error Handling and Messaging

Error Handler Flow Diagram

This flow diagram charts the possible paths an error might take once it has exited the general main ECO Business Object Interface process flow.

Error handling Concepts

Error handling depends on the severity of the error, the entities the error extends itself over (scope of the error) , and how the error affects the lineage (child record error states). When an error occurs, records are marked so that erroneous records are not processed again.

Error Severity Levels

Severity levels help distinguish between different types of errors since the import program behaves differently for each of these errors. The following is a list of the error severity levels recognized by the import program:

CODE	MEANING
'W'	Warning / Debug
'E'	Standard Error
'E'	Severe Error
'F'	Fatal Error
'U'	Unexpected error

Error States

Error states serve two purposes :

- They convey to the user the exact type of error in the record.
- They help the import program identify the records that do not need to be processed.

CODE	MEANING
'S'	Success
'E'	Error
'F'	Fatal Error

CODE	MEANING
'U'	Unexpected Error
'N'	Not Processed

Error Scope

This indicates what the depth of the error is in the business object, that is, how many other records in the business object hierarchy the current error affects.

CODE	MEANING
'R'	Error affects current 'R'ecord
'S'	Error affects all 'S'ibling and child records
'C'	Error affects 'C'hild records
'A'	Error affects 'A'll records in business object

Child Error States

If an error in a record affects child records, the status of the child may vary based on the type of error. There are two error states that indicate how the child is affected:

CODE	MEANING
'E'	Error
'N'	Not Processed

Error Classes

There are three major classes that determine the severity of the problem.

Expected errors: These are errors the program specifically looks for in the business object, before committing it to the production tables.

1. Standard Error: This error causes only the current record to be error-ed out, but is not serious enough to affect any other records in the object. The current record

status is set to 'E'. For example: Revised item cannot be updated to status=10.

2. Severe Error I: This error affects all records. All record statuses are set to 'E'. This error is usually a change notice/organization uniformity error. All records must have the same change notice/organization combination.

3. Severe Error I: This error affects all records. All record statuses are set to 'E'. This error is usually a change notice/organization uniformity error. All records must have the same change notice/organization combination.

4. Severe Error III: This error not only affects the current record but also its child records in the business object. The child record statuses are set to 'E'. Please check your child records for errors as well as the current record. This error is usually a user-unique to unique index conversion error.

5. Severe Error IV: This error affects the current record and its child records since the program cannot properly process the child records. The child record statuses are set to 'N'. This type of errors occur when there are errors on CREATEs.

6. Fatal Error I: These errors occur when it is not possible to perform any operation on the ECO. Such errors affect the entire business object. All record statuses are set to 'F'. The following are situations that cause this error:
   - ECO is implemented in the production tables
   - ECO is canceled in the production tables
   - Workflow activity is in progress for the ECO
   - You do not have access to this ECO (change order type)

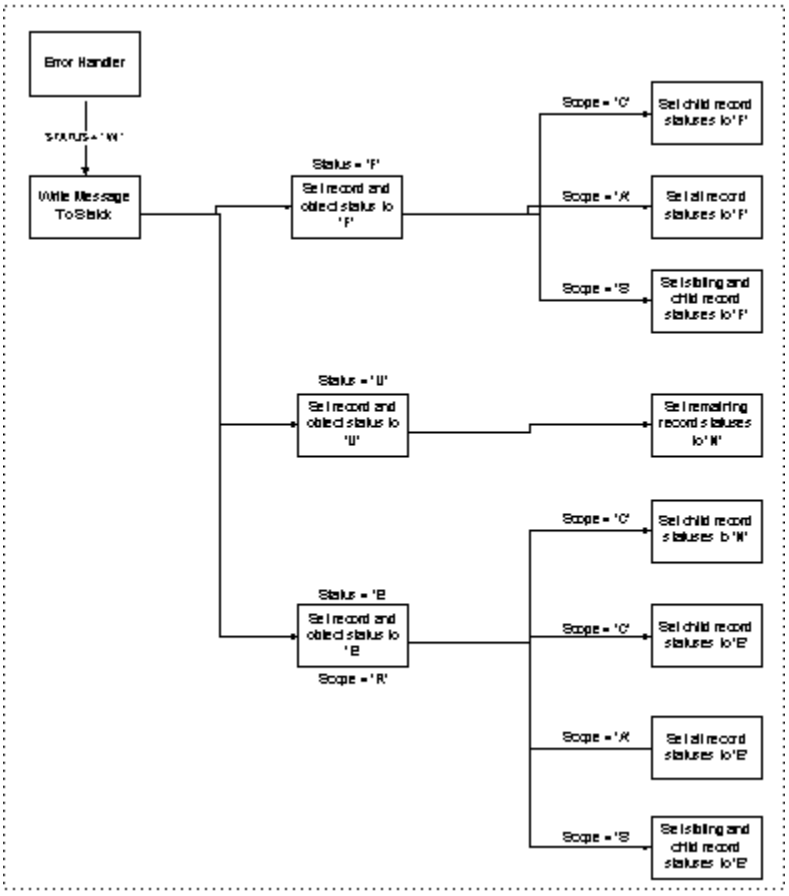
7. Fatal Error II: This error affects all records that are children of this record's parent, when the parent is not in the business object. A characteristic of this record's parent caused the error, so all its siblings and their children also get a record status of 'F'. This usually occurs when the user tries to create, update, or delete a component of a revised item that was already implemented or cancelled.

8. Fatal Error III: These errors affects the current record and its children, since it is not possible to perform any operation on these records. The current record and all its child record statuses are set to 'F'. The following situations cause this error:
   - Revised item is implemented in the production tables
   - Revised item is canceled in the production tables
   - You do not have access to this revised item (BOM item type)

Unexpected errors: All errors that are not expected errors are unexpected errors. These

are errors that the program is not specifically looking for, for example, the user somehow loses the database connection.

Unexpected Errors



Warnings: These are messages for information only. The purpose of warnings is:

1. to warn the user of problems that may occur when the ECO is implemented. For example: Revised item is being referenced on another pending ECO.

2. to inform the user of any side-effects caused by user-entered data. For example: All Approval History records associated with ECO have been deleted.

How it all works

To bring together all the concepts above into a workable algorithm, we must introduce some terms that used extensively in this section, and the rest of the document.

Child record : All records carry the unique keys for all parents above them. A child record (of a particular parent) is one that holds the same values for the unique key columns as the parent record.

Sibling record : A sibling record (of the current record) is one that holds the same parent unique key column values as the current record. For example, a component record that holds the same parent revised item unique key column values as the current component record, is a sibling of the current component record. Likewise, a reference designator record that holds the same parent component unique key column values as a substitute component is a sibling of the substitute component.

Business Object Error Status : This indicates the state of the whole business object after the import. As soon as the import program encounters an erroneous record, it sets the business object error status (also called return status) appropriately to convey this to the user. It is then up to the user to locate the offending record(s) using the individual record error statuses as indicated below. The caller may also use the business object return status to choose an appropriate course of action (commit, rollback, or further processing by the caller).

The following is a list of all the possible business object states.

CODE	MEANING
'S'	Success
'E'	Error
'F'	Fatal Error
'U'	Unexpected Error

Record Error Status : This is the state of the record after the import (success or error). The error status is also referred to as return status or error state in this document. Please see the Error States section above for a list of statuses a record can receive. The error status helps locate erroneous records in a business object that has error-ed out. The following are important pointers about the record error status.

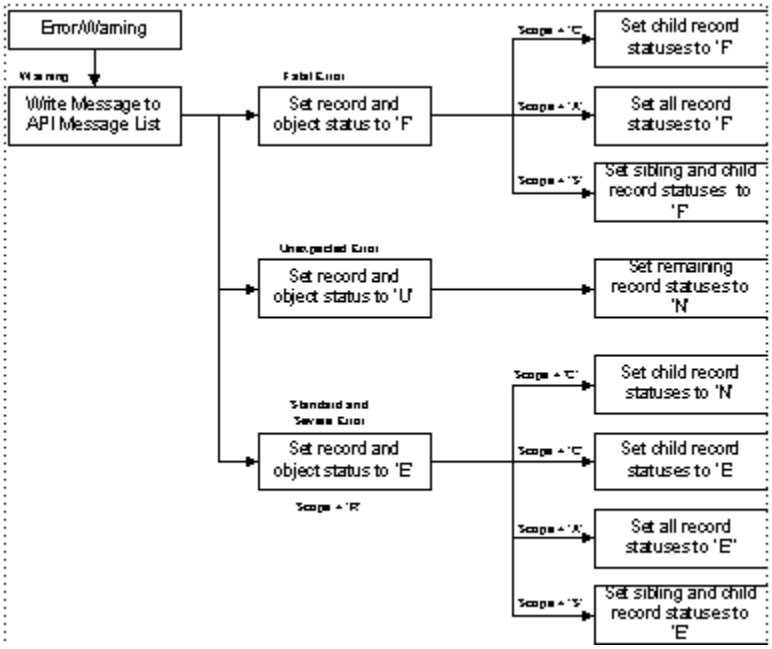
- Every record is assigned an error status by the import program. Hence, if a record has a NULL return status, it means that the import program has not gotten to it yet.
- The user must send in records with {return status = NULL}. The import program will not look at records that already have an error status value, since it assumes that a record with an error status value has already been looked at by the program.

The following shows how the error status, error scope, and child statuses relate together to constitute the different error classes for records

Error	Status	Scope	Child Statuses
Warning	S: Success	R: Record Only	-N/A-
Standard Error	E: Error	R: Record Only	-N/A-
Severe Error I	E: Error	A: All Records	E: Error
Severe Error II	E: Error	S: Current, Sibling and Child Records	E: Error
Severe Error III	E: Error	C: Current and Child Record	E: Error
Severe Error IV	E: Error	C: Current and Child Records	N: Not Processed
Fatal Error I	F: Fatal Error	A: All Records	
Fatal Error II	F: Fatal Error	S: Current, Sibling and Child Records	
Fatal Error III	F: Fatal Error	C: Current and Child Record	
Unexpected Error	U: Unexpected Error	-N/A-	N: Not Processed

This flow diagram charts the possible paths an error might take:

Possible Error Paths



The list below shows the sequence of steps that need to be performed when warnings or errors are encountered:

p_severity_level = Standard Error.

- Log error messages
- Set record status to 'E'
- Set business object status to 'E'

p_severity_level = Severe Error I.

- Log error messages
- Set record status to 'E'
- Set all business object record statuses to 'E'
- Set business object status to 'E'

p_severity_level = Severe Error II.

- Log error messages
- Set record status to 'E'
- Set direct and indirect children statuses to 'E'. Also set statuses of sibling records

and all their children to 'E'.

- Set business object status to 'E'

p_severity_level = Severe Error III.

- Log error messages
- Set record status to 'E'
- Set direct and indirect children statuses to 'E'.
- Set business object status to 'E'

p_severity_level = Severe Error IV.

- Log error messages
- Set record status to 'E'
- Set direct and indirect children statuses to 'N'.
- Set business object status to 'E'

p_severity_level = Fatal Error I.

- Log error messages
- Set record status to 'F'
- Set all business object records to 'F'.
- Set business object statuses to 'F'

p_severity_level = Fatal Error II.

- Log error messages
- Set record status to 'F'
- Set direct and indirect children statuses to 'F'. Also set statuses of sibling records and all their children to 'F'.
- Set business object statuses to 'F'

p_severity_level = Fatal Error III.

- Log error messages
- Set record status to 'F'

- Set direct and indirect children statuses to 'F'
- Set business object statuses to 'F'

p_severity_level = Unexpected Error.

- Log error messages
- Set record status to 'U'
- Set all remaining un-processes business object record statuses to 'N'.
- Set business object statuses to 'U'

p_severity_level = Warning.

- Log error messages

API Messaging

API Message Table

All messages are logged in the API Error Message Table. This is a PL/SQL table (array) of messages. Please see Accessing Messages in the Launching the Import section of this document on how to access these messages.

The following is a description of the API Message Table:

Field	Type	Description
Message_Text	VARCHAR2(2000)	The actual message that the user sees. Please see below for format information.
Entity_Id	VARCHAR2(3)	The entity that this message belongs to. This may hold BO, ECO, REV, RI, RC, RD, or SC.BO - Business ObjectECO - ECO HeaderREV - ECO RevisionsRI - Revised ItemsRC - Revised ComponentsRD - Reference DesignatorsSC - Substitute Components

Field	Type	Description
Entity_Index	NUMBER	The index of the entity array this record belongs to.
Message_Type	VARCHAR2(1)	Indicates whether message is an error or warning.

Message Formats

Expected errors and warnings: The message text contains the translated and token substituted message text. Please note that the message text may contain tokens, some of which will identify the entity instances that this message is for. The following tokens identify the several entities:

- Revised Item : Revised_Item_Name
- Revised Component : Revised_Component_Number
- Substitute Component : Substitute_Component_Number
- Reference Designator : Reference_Designator_Name
- ECO Revisions : Revision
- ECO Header : ECO_Name

Unexpected Errors

<Package Name> <Procedure/Function Name> <SQL Error Number>

<SQL Error Message Text>

Other Message

An Other Message is a message that is logged for all records that are affected by an error in a particular record. So if an error in a revised item record will cause all its children to error out, then the following will be logged:

- For the revised item itself, the error message describing the problem.
- For all records affected by the type of error that occurred in the revised item, the other message. This message essentially mentions the following:

1. how the error has affected this record, that is, it has been error-ed out too, with a

severe or fatal error status, ot that it has bit been processed.

2. which record caused this record to be affected.

3. what process flow step in the offending record caused this record to be affected.

4. what transaction type in the offending record caused this record to be affected.

Essentially the purpose of the other message is to give the user as much information as possible about the error that got propagated to this record.

Error Handler

The program performs all its error handling and messaging through the Error Handler. It makes calls to the Error Handler when an error or warning needs to be issued. The following are the functions of the Error Handler:

- Log the error/warning messages sent to it.
- Set the return status of the record in error.
- Set the return status of other records affected by this error.

The following is the input that the Error Handler expects from the calling program.

Input	Description
Business Object	Calling program must pass the whole business object as-is.
Message and Token List	List of messages generated for error in the current record. See below for description of this list.
Error Status	Status the record in error should be set to.
Error Level	Business Object hierarchy level that current record is an instance of. That is, the entity that the record in error belongs to.
Entity Array Index	Index of record in error in its encompassing entity array. Does not apply to ECO Header.
Error Scope	Indicates depth of error, that is, how many other records are affected by it.

Input	Description
Other Message and Token List	Message generated for the other affected records. See below for description.
Other Status	Status the other affected records should be set to.

Message and Token List Records

The Message and Token List, and the Other Message and Token List are temporary arrays that the calling program maintains. They hold message-and-token records. The Error Handler must log these messages into the API Message List. The calling program may want some of these message record tokens to be translated (such tokens are typically messages themselves).

For expected errors and warnings, the translated message text is retrieved using the message name. Then, after any requested token translation is performed, the tokens are substituted into the translated message text, and the message is logged. For unexpected errors, the calling program itself sends a message text, so no message retrieval is needed. The message is logged after token translation and substitution is performed.

Field	Description
Message Name	Name of the message used to retrieve the translated message text. NULL for unexpected errors.
Message Text	Message text for unexpected errors.
Token Name	Name of the token in the message.
Token Value	Value of the token in the message.
Translate	Should this token value be translated ?

Since each message may have more than one token, the Message and Token List has as many occurrences of the same message as there are tokens in it. The same applies to the Other Message and Token List, except that this list needs to carry only one message which is assigned to all other affected records. Since this lone message may have more than one token, there may be more than one occurrence of the message in the list.

Please note that the API Message List is public, but the Message and Token Lists are

not.

9

Oracle Flow Manufacturing Open Interfaces and APIs

This chapter covers the following topics:

- Flow Schedule API

Flow Schedule API

The Flow Schedule API is used to create, update, delete, retrieve, lock, schedule and unlink order lines of flow schedules.

Functional Overview

The Flow Schedule API provides five procedures for create, update, delete, retrieve, lock schedule and unlink order lines of flow schedules.

- Process_Flow_Schedule: To create, update, delete a flow schedule.
- Lock_Flow_Schedule: To lock a flow schedule row.
- Get_Flow_Schedule: To retrive a flow schedule from the table.
- Line_Schedule: To Schedule range of flow schedules in a date range using a given scheduling rule.
- Unlink_Order_line: To remove sales order reference in flow schedule.

Package Name: MRP_FLOW_SCHEDULE_PUB

Procedure Name: PROCESS_FLOW_SCHEDULE

The following table describes all parameters used by PROCESS_FLOW_SCHEDULE.

Parameter	Usage	Type	Required	Derived	Optional
p_version_number	IN	NUMBER	x	-	-
p_init_msg_list	IN	VARCHAR2	-	-	x
p_return_values	IN	VARCHAR2	-	-	x
p_commit	IN	VARCHAR2	-	-	x
x_return_status	OUT	VARCHAR2	-	-	-
x_msg_count	OUT	NUMBER	-	-	-
x_msg_data	OUT	VARCHAR2	-	-	-
p_flow_schedule_rec	IN	record	-	-	x
p_flow_schedule_val_rec	IN	record	-	-	x
x_flow_schedule_rec	OUT	record	-	-	-
x_flow_schedule_val_rec	OUT	record	-	-	-

p_version_number

API version number

p_flow_schedule_rec

Flow schedule record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_REC_TYPE. Default value: G_MISS_FLOW_SCHEDULE_VAL_REC

p_flow_scheduleval_rec

Flow schedule value record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_VAL_REC_TYPE. Default value: G_MISS_FLOW_SCHEDULE_VAL_REC

x_flow_schedule_rec

Flow schedule record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_REC_TYPE

x_flow_schedule_val_rec

Flow schedule value record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_VAL_REC_TYPE

The following table describes all parameters used by FLOW_SCHEDULE_REC_TYPE.

Parameter	Type	Default
alternate_bom_designator	VARCHAR2(10)	FND_API.G_MISS_CHAR
alternate_routing_desig	VARCHAR2(10)	FND_API_G_MISS_CHAR
attribute1-15	VARCHAR2(150)	FND_API_G_MISS_CHAR
attribute_category	VARCHAR2(30)	FND_API_G_MISS_CHAR
bom_revision	VARCHAR2(3)	FND_API_G_MISS_CHAR
bom_revision_date	DATE	FND_API_G_MISS_DATE
build_sequence	NUMBER	FND_API_G_MISS_NUM
class_code	VARCHAR2(10)	FND_API_G_MISS_CHAR
completion_locator_id	NUMBER	FND_API_G_MISS_NUM
completion_subinventory	VARCHAR2(10)	FND_API_G_MISS_CHAR
date_closed	DATE	FND_API_G_MISS_DATE
demand_class	VARCHAR2(30)	FND_API_G_MISS_CHAR
demand_source_delivery	VARCHAr2(30)	FND_API_G_MISS_CHAR
demand_source_header_id	NUMBER	FND_API_G_MISS_NUM
demand_source_line	VARCHAR2(30)	FND_API_G_MISS_CHAR

Parameter	Type	Default
demand_source_type	NUMBER	FND_API_G_MISS_NUM
line_id	NUMBER	FND_API_G_MISS_NUM
material_account	NUMBER	FND_API_G_MISS_NUM
material_overhead_account	NUMBER	FND_API_G_MISS_NUM
material_variance_account	NUMBER	FND_API_G_MISS_NUM
mps_net_quantity	NUMBER	FND_API_G_MISS_NUM
mps_scheduled_comp_date	DATE	FND_API_G_MISS_DATE
organization_id	NUMBER	FND_API_G_MISS_NUM
outside_processing_acct	NUMBER	FND_API_G_MISS_NUM
outside_proc_var_acct	NUMBER	FND_API_G_MISS_NUM
overhead_account	NUMBER	FND_API_G_MISS_NUM
overhead_variance_account	NUMBER	FND_API_G_MISS_NUM
planned_quantity	NUMBER	FND_API_G_MISS_NUM
primary_item_id	NUMBER	FND_API_G_MISS_NUM
project_id	NUMBER	FND_API_G_MISS_NUM
quantity_completed	NUMBER	FND_API_G_MISS_NUM
request_id	NUMBER	FND_API_G_MISS_NUM
resource_account	NUMBER	FND_API_G_MISS_NUM
resource_variance_account	NUMBER	FND_API_G_MISS_NUM
routing_revision	VARCHAR2(3)	FND_API_G_MISS_CHAR

Parameter	Type	Default
routing_revision_date	DATE	FND_API_G_MISS_DATE
scheduled_completion_date	DATE	FND_API_G_MISS_DATE
scheduled_flag	NUMBER	FND_API_G_MISS_NUM
scheduled_start_date	DATE	FND_API_G_MISS_DATE
schedule_group_id	NUMBER	FND_API_G_MISS_NUM
schedule_number	VARCHAR2(30)	FND_API_G_MISS_CHAR
status	NUMBER	FND_API_G_MISS_NUM
std_cost_adjustment_acct	NUMBER	FND_API_G_MISS_NUM
task_id	NUMBER	FND_API_G_MISS_NUM
wip_entity_id	NUMBER	FND_API_G_MISS_NUM
scheduled_by	NUMBER	FND_API_G_MISS_NUM
return_status	VARCHAR2(1)	FND_API_G_MISS_CHAR
db_flag	VARCHAR2(1)	FND_API_G_MISS_CHAR
operation	VARCHAR2(30)	FND_API_G_MISS_CHAR
end_item_unit_number	VARCHAR2(30)	FND_API_G_MISS_CHAR
quantity_scrapped	NUMBER	FND_API_G_MISS_NUM
kanban_card_id	NUMBER	FND_API_G_MISS_NUM
created_by	NUMBER	FND_API_G_MISS_NUM
creation_date	DATE	FND_API_G_MISS_DATE
last_updated_by	NUMBER	FND_API_G_MISS_NUM

Parameter	Type	Default
last_update_date	DATE	FND_API_G_MISS_DATE
last_update_login	NUMBER	FND_API_G_MISS_NUM
program_application_id	NUMBER	FND_API_G_MISS_NUM
program_id	NUMBER	FND_API_G_MISS_NUM
program_update_date	DATE	FND_API.G_MISS_DATE

The following table describes all parameters used by FLOW_SCHEDULE_VAL_REC_TYPE.

Parameter	Type	Default
completion_locator	VARCHAR2(240)	FND_API.G_MISS_CHAR
line	VARCHAR2(240)	FND_API.G_MISS_CHAR
organization	VARCHAR2(240)	FND_API.G_MISS_CHAR
primary_item	VARCHAR2(240)	FND_API.G_MISS_CHAR
project	VARCHAR2(240)	FND_API.G_MISS_CHAR
schedule_group	VARCHAR2(240)	FND_API.G_MISS_CHAR
task	VARCHAR2(240)	FND_API.G_MISS_CHAR
wip_entity	VARCHAR2(240)	FND_API.G_MISS_CHAR

Package Name: MRP_FLOW_SCHEDULE_PUB

Procedure Name: LOCK_FLOW_SCHEDULE

The following table describes all parameters used by LOCK_FLOW_SCHEDULE.

Parameter	Usage	Type	Required	Derived	Optional
p_version_number	IN	NUMBER	x	-	-
p_init_msg_list	IN	VARCHAR2	-	-	x
p_return_values	IN	VARCHAR2	-	-	x
p_commit	IN	VARCHAR2	-	-	x
x_return_status	OUT	VARCHAR2	-	-	-
x_msg_count	OUT	NUMBER	-	-	-
x_msg_data	OUT	VARCHAR2	-	-	-
p_flow_schedule_rec	IN	RECORD	-	-	x
p_flow_schedule_val_rec	IN	RECORD	-	-	x
x_flow_schedule_rec	OUT	RECORD	-	-	-
x_flow_schedule_val_rec	OUT	RECORD	-	-	-

p_version_number

API version number

p_flow_schedule_rec

Flow schedule record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_REC_TYPE. Default value: G_MISS_FLOW_SCHEDULE_VAL_REC

p_flow_scheduleval_rec

Flow schedule value record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_VAL_REC_TYPE. Default value: G_MISS_FLOW_SCHEDULE_VAL_REC

x_flow_schedule_rec

Flow schedule record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_REC_TYPE

x_flow_schedule_val_rec

Flow schedule value record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_VAL_REC_TYPE.

Package Name: MRP_FLOW_SCHEDULE_PUB

Procedure Name: GET_FLOW_SCHEDULE

The following table describes all parameters used by GET_FLOW_SCHEDULE.

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	NUMBER	x	-	-
p_init_msg_list	IN	VARCHAR2	-	-	x
p_return_values	IN	VARCHAR2	-	-	x
p_commit	IN	VARCHAR2	-	-	x
x_return_status	OUT	VARCHAR2	-	-	-
x_msg_count	OUT	NUMBER	-	-	-
x_msg_data	OUT	VARCHAR2	-	-	-
p_wip_entity_id	IN	NUMBER	x	-	-
p_wip_entity	IN	VARCHAR2	x	-	-
x_flow_schedule_rec	OUT	RECORD	-	-	-
x_flow_schedule_val_rec	OUT	RECORD	-	-	-

p_api_version_number

API version number.

p_init_msg_list

Standard Oracle API parameter.

p_return_values

Standard Oracle Output parameter.

p_commit

Standard Oracle API parameter.

x_return_status

Standard Oracle API Output parameter.

x_msg_count

Standard Oracle API Output parameter.

x_msg_data

Standard Oracle API Output parameter.

p_wip_entity_id

Standard Oracle API Output parameter.

p_wip_entity

Standard Oracle API Output parameter.

x_flow_schedule_rec

Flow schedule record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_REC_TYPE.

x_flow_schedule_val_rec

Flow schedule value record type. Record type: MRP_FLOW_SCHEDULE_PUB. FLOW_SCHEDULE_VAL_REC_TYPE.

Package Name: MRP_FLOW_SCHEDULE_PUB

Procedure Name: LINE_SCHEDULE

The line_schedule API will reschedule the flow schedules in the given line, organization, and completion date range using the scheduling rule. This API only schedules the flow schedules created in the same session as the creation of the original flow schedule which is created by PROCESS_FLOW_SCHEDULE API. The following

table provides the specifications for this API:

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	NUMBER	x	-	-
x_return_status	OUT	VARCHAR2	-	-	x
x_msg_count	OUT	NUMBER	-	-	x
x_msg_data	OUT	VARCHAR2	-	-	x
p_rule_id	IN	NUMBER	x	-	-
p_line_id	IN	NUMBER	x	-	-
p_org_id	IN	NUMBER	x	-	-
p_sched_start_date	IN	DATE	x	-	-
p_sched_end_date	IN	DATE	x	-	-
p_update	IN	NUMBER	x	-	-

p_rule_id

Line scheduling rule id. It indicates the scheduling rule to be used to scheduling the flow schedule.

p_line_id

Sales order Line identifier.

p_org_id

Organization id.

p_sched_start_date

The start date of the flow schedules to be scheduled.

p_sched_end_date

The end date of the flow schedules to be scheduled.

Package Name: MRP_FLOW_SCHEDULE_PUB

Procedure Name: UNLINK_ORDER_LINE

The unlink_order_line API will remove a given sale order reference from all flow schedules that make reference to the given sales order. The following table provides the specifications for this API:

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	NUMBER	x	-	-
x_return_status	OUT	VARCHAR2	-	-	x
x_msg_count	OUT	NUMBER	-	-	x
x_msg_data	OUT	VARCHAR2	-	-	x
p_assembly_item_id	IN	NUMBER	x	-	-
p_line_id	IN	NUMBER	x	-	-

p_assembly_item_id

Assembly item identifier.

p_line_id

Sales order Line identifier.

p_sched_end_date

The end date of the flow schedules to be scheduled.

10

Oracle Shop Floor Management Open Interfaces

This chapter covers the following topics:

- Import Lot Jobs Concurrent Program
- Lot Move Transactions Concurrent Program
- Import WIP Lot Transactions Concurrent Program
- Inventory Lot Transactions Interface Concurrent Program
- Lot Attributes

Import Lot Jobs Concurrent Program

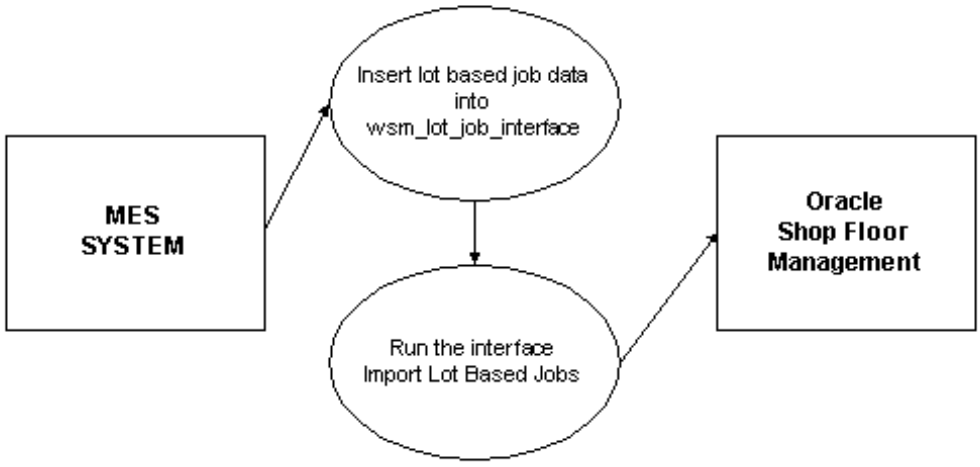
You can insert records into the import jobs interface from any manufacturing execution system or any other third party planning system.

Import Lot Based Jobs interface has two major components. The first one relates to the import of lot based jobs for the first sector, while the second component relates to import/creation of lot jobs for subsequent sectors. Once the lot jobs are imported successfully, you can use the lot based job forms to view the job and carry out subsequent move transactions through a move transaction form or move interface.

Import lot based jobs are processed in parallel based on user-configurable parameters set at installation. There are two profile options that can be adjusted according to capability and workload: Maximum number of import lot job workers, and maximum number of rows processed by a lot based job worker as a batch. Maximum number of import lot job workers sets the number of worker programs launched in parallel that process rows from wsm_lot_job_interface table. Maximum number of rows processed by a lot based job worker as a batch specifies the number of rows from the interface table each worker can process at a time. The values for these two options are typically set by the database administrator. When specifying values for these options, the total available memory and the load profile of the system should be taken into account. These profile options allow fine tuning to optimize processing and reduce the total

execution time depending on system processor availability and memory capability.

The following diagram describes the basic flow of the Import Lot Jobs Concurrent Program. The lot based job data from MES is inserted into the WSM_LOT_JOB_INTERFACE table by running a script. The Import Lot Based Job program validates the data and then imports the lot based job data from MES into Oracle Applications.



Import Lot Jobs Concurrent Program Features

There are two modes of lot based jobs, one for the first sector and one for the other subsequent sectors. The two modes are discriminated by the MODE_FLAG value. Creation of a lot for the first sector is indicated by a MODE_FLAG value of 1. This lot is released and moves through a series of operations to an inventory location, where it becomes an Inventory Lot. The Inventory Lot then moves into the next BOM level where it becomes a primary component for a new WIP lot in the next sector. MODE_FLAG has a value 2 when a WIP lot is created from an Inventory lot under these circumstances.

All these types of transactions create records in the existing WIP tables. A mode 1 job creates records into the following WIP base tables: WIP_DISCRETE_JOBS, WIP_ENTITIES, WIP_OPERATION_RESOURCES, WIP_REQUIREMENT_OPERATIONS, WIP_OPERATIONS, WIP_OPERATIONAL_YIELDS, and, if the job is in released status, WIP_PERIOD_BALANCES. A mode 2 job, additionally, creates rows into MTL_MATERIAL_TRANSACTIONS.

When the assembly of a lot-based job is serial controlled, rows are inserted/updated in MTL_SERIAL_NUMBERS table when serial numbers are generated/updated.

A site level profile "WSM: Create Job Level BOM and Routing Copies" is used to control the behavior of the concurrent program. The profile is always set to YES, the APS or

user can push detailed recommendations through the interface table WSM_LOT_JOB_DTL_INTERFACE, Shop Floor Management then run concurrent program "Import Lot Based Job" to import the job details information into Oracle application. The recommendations will be marked on the job level copy tables: WSM_COPY_OP_NETWORKS, WSM_COPY_OPERATIONS, WSM_COPY_REQUIREMENT_OPS, WSM_COPY_OP_RESOURCES, WSM_COPY_OP_RESOURCE_USAGE, WSM_COPY_OP_RESOURCE_INSTANCES.

Serial Support in Lot jobs concurrent program

The following features are supported for serial controlled assembly of a lot based job through lot jobs concurrent program:

- Associate existing serial numbers for a lot based job.
- Generate and associate new serials to a lot based job.
- Disassociate serial numbers from a lot based job.
- Specify serial attributes information for the associated serial numbers.

The following feature is not supported through LBJ Interface:

- Import lot based job from a serial control component inventory lot.

The Import lot jobs concurrent program processes Serial Number details by linking the data specified in WSM_LOT_JOB_INTERFACE and WSM_SERIALTXN_INTERFACE tables.

Functional Overview

1. For creating a mode 1 job, insert a row into WSM_LOT_JOB_INTERFACE. For creating a mode 2 job, additionally, insert a row into WSM_STARTING_LOTS_INTERFACE table. For a mode 2 job, the HEADER_ID of the row in WSM_STARTING_LOTS_INTERFACE table should be equal to the SOURCE_LINE_ID of the corresponding row in WSM_LOT_JOB_INTERFACE. For a mode 1 job, SOURCE_LINE_ID will have NULL

2. While populating the interface table, take care that the HEADER_ID column is populated using some sequence. HEADER_ID is a NOT NULL column and it should be unique. The processor code uses the HEADER_ID as a unique identifier for each row in the interface table.

3. The processor selects only rows from WLJI which are in the Pending status. All these rows are internally assigned GROUP_IDs and assigned to workers launched by the import program. If you specify a GROUP_ID when launching the concurrent program, only rows with given GROUP_ID will be picked up, and the concurrent

program will only launch a single worker to process these rows.

4. When the profile WSM: Create Job Level BOM and Routing Copies is set to YES (option 2), Shop Floor Management accepts APS or user's recommendations through the open interface. Shop Floor Management makes a job-level copy of the assembly's BOM and Routing whenever a lot based job is created and marks the recommendations on the copy according to the data populated in WSM_LOT_JOB_DTL_INTERFACE table.

5. In WSM_SERIALTXN_INTERFACE table, populate columns GENERATE_SERIAL_NUMBER and GENERATE_FOR_QTY with appropriate values to generate and associate serials. While dealing with existing serials, populate columns SERIAL_NUMBER and ACTION_FLAG with appropriate values to associate/update/disassociate the serials.

6. For information related to serial attributes, please refer **Support for Serial Attributes through Interfaces**, page 10-4.

Errors and Validations

1. Errors and warnings, if any, are written into the WSM_INTERFACE_ERRORS table. Appropriate messages are also given in the interface tables to refer to the WSM_INTERFACE_ERRORS table when there is any error.

2. In case of any error in serial processing, then the corresponding parent record in WSM_LOT_JOB_INTERFACE table is also errored out and the error message is written into WSM_INTERFACE_ERRORS table.

Support for Serial Attributes through Interfaces

Serial attributes behavior is similar in Import Lot Jobs, Lot Move Transactions and WIP Lot Transactions Concurrent Program.

- Through interface specifying NULL indicates that the user wants to retain the existing serial attributes.

- Specifying FND_API.G_NULL_NUM (for NUMBER data type fields), FND_API.G_NULL_CHAR (for VARCHAR2 data type fields) and FND_API.G_NULL_DATE (for DATE data type fields) indicates that the user intends to nullify the attributes in those columns.

- Unlike Lot attributes, attributes for serials do not have separate table and WSM_SERIAL_TXN_INTERFACE itself is used for entering and updating attributes.

- Serial attribute fields are not honored when GENERATE_SERIAL_NUMBER column is populated with 1, that is, while generating new serials and associating to a job.

Calling the Import Lot Based Jobs Concurrent Program

In Oracle Shop Floor Management, from the Run Requests menu, select Import Lot Based Jobs.

Validation of Import Lot Based Jobs

The following describes validation of importing lot-based jobs:

Standard

Oracle Shop Floor Management validates all required columns in the Import Lot Based Jobs Concurrent Program.

Error Handling

If any validation fails, that particular row is errored out. The Import Lot Based Jobs Concurrent Program processes the rows and reports the following values for every record.

Error Handling Table

Condition	PROCESS_STATUS	ERROR_MESSAGE
Failure	3	actual error message
Running	2	null
New Record	1	null

Load type of 5 should be used when creating a new lot based job.

Load type 6 indicates that an update will be performed on an existing job. The following update scenarios are currently supported:

1. Update of status of existing jobs.
2. Update of quantity of non-transacted jobs.
3. Rescheduling suggestions on start and end dates will be supported.
4. Update of co-product supply flag is supported.
5. For unreleased standard lot based job, updating alternate BOM and routing designator is supported.
6. For unreleased non-standard lot based job, updating BOM and routing reference is supported.

WSM_LOT_JOB_ INTERFACE Table

The following table lists the columns in the WSM_LOT_JOB_ INTERFACE table and provides their load/update type and validation information. Some columns are not used by the program and marked as ignored.

WSM_LOT_JOB_INTERFACE (WLJI) Table

Column	Type	Required	Derived	Optional	Permitted Values
mode_flag	number	x			1 - Create Lot based Job 2 - lot created from an inventory lot
last_update_date	date	x			
last_updated_by	number	x	Fnd_global.user_id	x	
creation date	date	x			
created_by	number	x	Fnd_global.user_id	x	
last_update_login	number			x	
request_id	number		fnd_global.conc_request_id		
program_id	number		fnd_global.comc_program_id		
program_application_id	number		fnd_global.prog_appl_id		
program_update_date	date			x	

Column	Type	Required	Derived	Optional	Permitted Values
group_id	number		wsm_lot_job_interface_s		
source_code	varchar2(30)			x	
source_line_id	number	required for mode_flag 2			
process_type	number				Ignored
organization_id	number	x			
load_type	number	x			5 - Job creation 6 - Job update
status_type	number	x			1 - Unreleased 3 - Released 4 - Complete 6 - On hold 7 - Canceled
old_status_type	number				Ignored
last_unit_completion_date	date	x			This column or the first_unit_completion_date are required.
old_completion_date	date				Ignored
processing_work_days	number				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
daily_production_rate	number				Ignored
line_id	number				Ignored
primary_item_id	number	x			
bom_reference_id	number			x	
routing_reference_id	number			x	
bom_revision_date	date			x	
routing_revision_date	date			x	
wip_supply_type	number				1 - Push 2 - Assembly Pull 3 - Operation Pull 4 - Bulk 5 - Vendor 6 - Based on BOM
class_code	varchar2(10)			x	
lot_number	varchar2(30)			x	
lot_control_code	number				Ignored
job_name	varchar2(240)			x	

Column	Type	Required	Derived	Optional	Permitted Values
description	varchar2(240)			x	
firm_planned_flag	number			x	1 - Yes 2 - No
alternate_routing_designator	varchar2(10)	x			
alternate_bom_designator	varchar2(10)	x			
demand_class	varchar2(30)				Ignored
start_quantity	number	x			
old_start_quantity	number				Ignored
wip_entity_id	number			x (for load type 6)	
repetitive_schedule_id	number				Ignored
error	varchar2(2000)		x		Ignored
parent_group_id	number				Ignored
attribute_category	varchar2(30)			x	
attribute1 - 15	varchar2(150)			x	
last_updated_by_name	varchar2(100)				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
created_by_name	varchar2(100)				Ignored
process_phase	number				Ignored
process_status	number	x			1 - Pending
organization_code	varchar2(3)		x		
first_unit_start_date	date	x			This column or the last_unit_completion_date are required.
first_unit_completion_date	date				Ignored
last_unit_start_date	date				Ignored
scheduling method	number	x			1 - Routing base (only supported when the profile is set to YES), infinite scheduler will be called to schedule the current job 2 - Lead time 3 - Manual
line_code	varchar2(10)				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
primary_item_segments	varchar2(2000)				Ignored
bom_reference_segments	varchar2(2000)			x	Ignored
routing_reference_segments	varchar2(2000)				Ignored
routing_revision	varchar2(3)			x	
bom_revision	varchar2(3)			x	
completion_subinventory	varchar2(10)			x	
completion_locator_id	number			x	
completion_locator_segments	varchar2(2000)				Ignored
schedule_group_id	number			x	
schedule_group_name	varchar2(30)			x	
build_sequence	number			x	
project_id	number				Ignored
project_name	varchar2(30)				Ignored
task_id	number				Ignored
task_name	varchar2(20)				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
net_quantity	number			x	
descriptive_flex_segments	varchar2(2000)			x	
project_number	varchar2(25)				Ignored
task_number	varchar2(25)				Ignored
project_costed	number				Ignored
end_item_unit_number	varchar2(30)				Ignored
overcompletion_tolerance_type	number				Ignored
overcompletion_tolerance_value	number				Ignored
kanban_card_id	number				Ignored
priority	number				Ignored
due_date	date				
allow_explosion	varchar2(1)	x			
header_id	number	x			
delivery_id	number				Ignored
error_code	number		x		

Column	Type	Required	Derived	Optional	Permitted Values
error_msg	varchar2(2000)		x		
interface_id	number				Ignored
transaction_date	date				If this column is given, only rows with transaction_date < SYSDATE will be pick up for processing
num_of_children	number		x		User input will be ignored

Control Columns

The following columns are control columns for the Import Lot Based Jobs program.

- ALLOW_EXPLOSION: This should always be set to Y to ensure that the bill of material and the routing of the assembly specified are exploded, else error is reported.
- HEADER_ID: Identifies each individual row in the table. Should have a unique value for each row.
- LOAD_TYPE: Determines whether the current interface record is to create a new lot-based job or to update an existing job. It also controls whether interface table columns are Required, Optional, Optional/Derived if Null, or Derived or Ignored, and has a NOT NULL restriction. You must assign one of the following possible values or an error occurs:
  - 5 Create Lot Based Job
  - 6 Reschedule Lot Based Job
- PROCESS_STATUS: The PROCESS_STATUS column indicates the current status of each record. Possible PROCESS_STATUS values include:

- 1 Pending
- 2 Running
- 3 Error

  Records should be inserted into the WSM_LOT_JOB_INTERFACE table with a PROCESS_STATUS = 1 (Pending). These values indicate that the record is ready to be processed. If the program fails at any stage when processing a record, the PROCESS_STATUS of that record is set to 3 (Error).

- STATUS TYPE: You can only create lot based jobs with a status of 3, Released or 1, Unreleased.
- Group_id: When launching 'Import Lot Based Job', you can specify group_id. Only rows with the given group_id will be picked up by the concurrent program.

Required Columns

You must specify values for columns in this category. If you do not enter a required value, the Import Lot Based Jobs program does not process the record and inserts an error record in the WSM_INTERFACE_ERRORS table for the appropriate record. Sometime it is not possible to write a row into WSM_LOT_JOB_INTERFACE table without specifying values for certain columns. For attributes that have a name and an associated id, if you specify values for both the name and the ID, the value for the ID is used and the value for the name is ignored during validation. If the entered or derived ID is NULL, you receive an error.

Optional Columns

You do not have to enter values for columns in this category.

Derived or Ignored Columns

These columns are for internal processing only. You should leave all columns in this category blank (NULL), since values entered in these columns are ignored or overwritten.

WSM_LOT_JOB_ DTL_INTERFACE Table

The following table lists the columns in the WSM_LOT_JOB_ DTL_INTERFACE table and provides their import operation/component and resource information. Some columns are not used by the program and marked as ignored.

WSM_LOT_JOB_DTL_INTERFACE (WLJDI) Table

Column	Type	Not Null	Description
record_id	number	x	Primary key of the table, value must be unique
interface_id	number		Interface identifier
group_id	number		Value in this column should be the same as the group_id in WSM_Lot_Job_interface table, if the details pertain a header record that is loaded at the same time
parent_header_id	number		This column will contain the header_id of the work order record (group_id, header_id columns from WSM_Lot_Job_Interface table will identify the header record uniquely). This column should be NULL if only the detail records are being loaed/updated

Column	Type	Not Null	Description
load_type	number	x	1 - Loading a resource 2 - Loading a component 3 - Loading an operation 4 - Loading a resource usage 5 - Loading a path link (To load a path, both routing_op_seq_num and next_routing_op_seq_num are required) 6 - Loading a path link and an operation (This is a combination of 3 and 5, if only operation info is loaded, next_routing_op_seq_num should be null) 7 - Loading a resource instance
substitution_type	number	x	1 - Delete 2 - Add 3 - Change 4 - Recommend and update 5 - Discommend (For path only, enabling user to specify small changes on the recommended path)

Column	Type	Not Null	Description
process_phase	number	x	Process phase
process_status	number	x	1 - Pending 2 - Running 3 - Error 4 - Completed 5 - Warning
job_name	varchar2(240)		Lot based job name
wip_entity_id	number		
organization_id	number		
organization_code	varchar2(3)		
department_id	number		Department Identifier
department_code	varchar2(4)		Department Code
description	varchar2(240)		Long description
job_op_seq_num	number		Value for this column is from operation_sequence_num in WIP_OPERATIONS table. It used to send recommendations for the current operation. The recommendation will be marked on the job's execution table, namely WO, WOR, WRO etc.

Column	Type	Not Null	Description
routing_op_seq_num	number		This should be the operation_sequence_num in WSM_COPY_OPERATIONS table. The recommendation will be marked on the job's copy table, namely WCO, WCOR, WCRO etc.
next_routing_op_seq_num	number		This is only required when a link is sent with load_type=5 or 6
resource_seq_num	number		Resource sequence number
resource_id_old	number		Resource identifier existing in current jobs
resource_id_new	number		For future operations, it must be the resource in WCOR (defined as either the primary or substitute resource of an operation in BOM). For current operation, it should be a valid resource defined in WIP_OPERATION_RESOURCES or WIP_SUB_OPERATION_RESOURCES.
resource_code_old	varchar2(10)		Resource code existing in current jobs
resource_code_new	varchar2(10)		Resource code to be added, changed or recommended

Column	Type	Not Null	Description
usage_rate_or_amount	number		Rate per assembly or amount per job
scheduled_flag	number		Method of scheduling the resource
assigned_units	number		Number of resources assigned to work
applied_resource_units	number		Amount of resource units charged
applied_resource_value	number		Value of resource units charged
uom_code	varchar2(3)		Unit of measure code
basis_type	number		Basis for scheduling and charging resource
activity_id	number		Activity identifier
autocharge_type	number		Method of charging the resource
standard_rate_flag	number		Indicates whether the resource is charged at the standard rate
start_date	date		Required for loading/updating a resource usage. If the usage block is specified then the start date pertains to that usage block

Column	Type	Not Null	Description
completion_date	date		Required for loading/updating a resource usage. If the usage block is specified then the completion date pertains to that usage block
schedule_seq_num	number		Schedule sequence number
substitute_group_num	number		Substitute group number of the resource
replacement_group_num	number		Replacement group number of the resource
setup_id	number		Setup identifier
batch_id	number		Batch identifier
inventory_item_id_old	number		Old material requirement identifier
inventory_item_id_new	number		For future operations, it must be the component in WCRO (defined as either the primary or substitute component of an operation in BOM). For current operation, it should be a valid component defined in WIP_REQUIREMENT_OPERATIONS.
inventory_item_new	varchar2(150)		New material requirement to be added, modified and recommended

Column	Type	Not Null	Description
inventory_item_old	varchar2(150)		Old material requirement
primary_item_id	number		The primary item identifier defined in BOM
primary_item	varchar2(150)		The primary item defined in BOM
src_phantom_item_id	number		The top phantom item identifier from which, this item comes
src_phantom_item	varchar2(150)		The top phantom item from which, this item comes
component_sequence_id	varchar2(150)		Component sequence identifier
quantity_per_assembly	number		Part usage quantity
wip_supply_type	number		Method of material consumption within WIP
date_required	date		Operation requirement start date
required_quantity	number		Part quantity required
quantity_issued	number		Part quantity issued
supply_subinventory	varchar2(10)		Subinventory used to supply component to WIP

Column	Type	Not Null	Description
supply_locator_id	number		Locator used to supply component to WIP
mrp_net_flag	number		Indicates whether or not MRP should consider the component requirement in its netting process
mps_required_quantity	number		Quantity used by MPS relief process
mps_date_required	date		Date used by MPS relief process
standard_operation_id	number		Standard operation identifier
scheduled_quantity	number		Scheduled quantity for the operation
operation_yeild	number		Operation yield
operation_start_date	date		Operation start date
operation_completion_date	date		Operation completion date
minimum_transfer_quantity	number		Minimum operation transfer quantity
backflush_flag	number		Backflush control code
count_point_type	number		Count point type
planning_pct	number		Network path planning percentage

Column	Type	Not Null	Description
transaction_date	date	x	Indicating the date of transaction, only records transaction_date > sysdate are picked up for processing.
last_update_date	date	x	Standard Who column - date when a user last updated this row
last_updated_by_name	varchar2(100)		Standard who column - user who last updated this row
last_updated_by	number		Standard who column - user identifier who last updated this row (foreign key to FND_USER.USER_ID)
creation_date	date	x	Standard who column - date when this row was created
created_by_name	varchar2(100)	x	Standard who column - user who created this row (foreign key to FND_USER.USER_ID)
created_by	number	x	Standard who column - user identifier who created this row
last_update_login	number		Standard who column

Column	Type	Not Null	Description
request_id	number		Concurrent Program who column - concurrent request id of the program that last updated this row (foreign key to FND_CONCURRENT_REQUESTS.REQUEST_ID)
program_application_id	number		Concurrent Program who column - application id of the program that last updated this row (foreign key to FND_APPLICATION.APPLICATION_ID)
program_id	number		Concurrent Program who column - program id of the program that last updated this row (foreign key to FND_CONCURRENT_PROGRAM.CONCURRENT_PROGRAM_ID)
program_update_date	date		Concurrent Program who column - date when a program last updated this row)
attribute_category	varchar2(30)		Descriptive flexfield structure definition column
attribute1	varchar2(150)		Descriptive flexfield segment
attribute2	varchar2(150)		Descriptive flexfield segment

Column	Type	Not Null	Description
attribute3	varchar2(150)		Descriptive flexfield segment
attribute4	varchar2(150)		Descriptive flexfield segment
attribute5	varchar2(150)		Descriptive flexfield segment
attribute6	varchar2(150)		Descriptive flexfield segment
attribute7	varchar2(150)		Descriptive flexfield segment
attribute8	varchar2(150)		Descriptive flexfield segment
attribute9	varchar2(150)		Descriptive flexfield segment
attribute10	varchar2(150)		Descriptive flexfield segment
attribute11	varchar2(150)		Descriptive flexfield segment
attribute12	varchar2(150)		Descriptive flexfield segment
attribute13	varchar2(150)		Descriptive flexfield segment
attribute14	varchar2(150)		Descriptive flexfield segment
attribute15	varchar2(150)		Descriptive flexfield segment
instance_id_new	varchar2(30)		Instance Identifier to be added, changed, recommended

Column	Type	Not Null	Description
instance_id_old	varchar2(30)		Instance Identifier that is in the job
serial_number_new	varchar2(30)		Serial number to be added, changed, recommended
serial_number_old	varchar2(30)		Serial number that is in the job
auto_request_material	varchar2(1)		Automatic request material
error_code	NUMBER		Error code
error_msg	VARCHAR2(2000)		Error message
comments	varchar2(240)		Comments

Validation For WSM_LOT_JOB_DTL_INTERFACE

The following tables describe validation for WSM_LOT_JOB_DTL_INTERFACE

For Loading a Link

The following table describes column validation for the WSM_LOT_JOB_DTL_INTERFACE table for loading a link:

Column Name	Load Type=5(Load a Link) Substitution Type=4(Recommend)	Load Type=5(Load a Link) Substitution Type=5(Discommend)	Load Type=6(Link and/or Op) Substitution Type=4(Recommend)
record_id	required	required	required
interface_id			
group_id	required	required	required
parent_header_id	optional	optional	optional

Column Name	Load Type=5(Load a Link) Substitution Type=4(Recommend)	Load Type=5(Load a Link) Substitution Type=5(Discommend)	Load Type=6(Link and/or Op) Substitution Type=4(Recommend)
load_type	5 (required)	5 (required)	6 (required)
substitution_type	4 (required)	5 (required)	4 (required)
process_phase			
process_status	1 (required)	1 (required)	1 (required)
job_name	optional	optional	optional
wip_entity_id	optional	optional	optional
organization_id	optional	optional	optional
organization_code	optional	optional	optional
department_id	optional	optional	optional
department_code	optional	optional	optional
description			
job_op_seq_num			
routing_op_seq_num	required	required	required
next_routing_op_seq_num	required	required	required
transaction_date	required	required	required
last_update_date	required	required	required
last_update_by	optional	optional	optional
last_update_by_name	optional	optional	optional

Column Name	Load Type=5(Load a Link) Substitution Type=4(Recommend)	Load Type=5(Load a Link) Substitution Type=5(Discommend)	Load Type=6(Link and/or Op) Substitution Type=4(Recommend)
creation_date	required	required	required
created_by	required	required	required
created_by_name	required	required	required
last_update_login	optional	optional	optional

Note: If parent_header_id is not given, wip_entity_id is required.

For Loading an Operation

The following table describe column validation for WSM_LOT_JOB_DTL_INTERFACE table for loading an operation:

Column Name	Load Type=3 (Operation) Substitution Type=3 or 4(Update)	Load Type=6(Link and/or Op) Substitution Type=3 or 4(Update)
record_id	required	required
interface_id		
group_id	required	required
parent_header_id	optional	optional
load_type	3 (required)	6 (required)
substitution_type	3,4 (required)	3,4 (required)
process_phase		
process_status	1 (required)	1 (required)

Column Name	Load Type=3 (Operation) Substitution Type=3 or 4(Update)	Load Type=6(Link and/or Op) Substitution Type=3 or 4(Update)
job_name	optional	optional
wip_entity_id	optional	optional
organization_id	optional	optional
organization_code	optional	optional
department_id	optional	optional
department_code	optional	optional
description	null	null
job_op_seq_num	required (current op)	required (current op)
routing_op_seq_num	required (future op)	required (future op)
next_routing_op_seq_num		
standard_operation_id		
scheduled_quantity	optional	optional
operation_yield	optional (future op)	optional (future op)
operation_start_date	optional	optional
operation_completion_date	optional	optional
minimum_transfer_quantity		
backflush_flag		
count_point_type		
last_update_date	required	required

Column Name	Load Type=3 (Operation) Substitution Type=3 or 4(Update)	Load Type=6(Link and/or Op) Substitution Type=3 or 4(Update)
last_update_by	optional	optional
last_update_by_name	optional	optional
creation_date	required	required
created_by	required	required
created_by_name	required	required
last_update_login	optional	optional
transaction_date	required	required

Note: If parent_header_id is not given, wip_entity_id is required.

For Loading a Resource

The following table describe column validation for WSM_LOT_JOB_DTL_INTERFACE table for loading a resource:

Column Name	Load Type=1(Resource), Substitution Type=4(Recom.)	Load Type=1(Resource), Substitution Type=3(Change)
record_id	required	required
interface_id		
group_id	required	required
parent_header_id	optional	optional
load_type	1 (required)	1 (required)
substitution_type	4 (required)	3 (required)

Column Name	Load Type=1(Resource), Substitution Type=4(Recom.)	Load Type=1(Resource), Substitution Type=3(Change)
process_phase		
process_status	1 (required)	1 (required)
job_name	optional	optional
wip_entity_id	optional	optional
organization_id	optional	optional
organization_code	optional	optional
department_id	optional	optional
department_code	optional	optional
description		
job_op_seq_num	required (current op)	required (current op)
routing_op_seq_num	required (future op)	required (future op)
next_routing_op_seq_num	null	null
resource_seq_num	required	required
resource_id_old		
resource_id_new	required	required
resource_code_old		
resource_code_new	optional	optional
uasge_rate_or_amount		
scheduled_flag		

Column Name	Load Type=1(Resource), Substitution Type=4(Recom.)	Load Type=1(Resource), Substitution Type=3(Change)
assigned_units		
applied_resource_units		
applied_resource_value		
uom_code		
basic_type		
activity_id		
autocharge_type		
standard_rate_flag		
start_date	optional	optional
completion_date	optional	optional
schedule_seg_num		
substitute_group_num	required	required
replacement_group_num	required	required
setup_id		
batch_id		
transaction_date	required	required
last_update_date	required	required
last_update_by	optional	optional
last_update_by_name	optional	optional

Column Name	Load Type=1(Resource), Substitution Type=4(Recom.)	Load Type=1(Resource), Substitution Type=3(Change)
creation_date	required	required
created_by	required	required
created_by_name	required	required
last_update_login	optional	optional

Note: If parent_header_id is not given, wip_entity_id is required.

For Loading a Resource Instance

The following table describe column validation for WSM_LOT_JOB_DTL_INTERFACE table for loading a resource instance:

Column Name	Load Type=7(Resource Instance) Substitution Type=4(Recom.)	Load Type=7(Resource Instance) Substitution Type=2(Add.)	Load Type=7(Resource Instance) Substitution Type=1(Delete.)
record_id	required	required	required
interface_id			
group_id	required	required	required
parent_header_id	optional	optional	optional
load_type	7 (required)	7 (required)	7 (required)
substitution_type	4 (required)	2 (required)	1 (required)
process_phase			
process_status	1 (required)	1 (required)	1 (required)

Column Name	Load Type=7(Resource Instance) Substitution Type=4(Recom.)	Load Type=7(Resource Instance) Substitution Type=2(Add.)	Load Type=7(Resource Instance) Substitution Type=1(Delete.)
job_name	optional	optional	optional
wip_entity_id	optional	optional	optional
organization_id	optional	optional	optional
organization_code	optional	optional	optional
department_id	optional	optional	optional
department_code	optional	optional	optional
description			
job_op_seq_num	required (current op)	required (current op)	required (current op)
routing_op_seq_num	required (future op)	required (future op)	required (future op)
next_routing_op_seq_num	required	required	required
resource_seq_num	required	required	required
resource_id_old			
resource_id_new	optional	optional	
resource_code_old			
resource_code_new	optional	optional	
usage_rate_or_amount			
scheduled_flag			
assigned_units	required	required	

Column Name	Load Type=7(Resource Instance) Substitution Type=4(Recom.)	Load Type=7(Resource Instance) Substitution Type=2(Add.)	Load Type=7(Resource Instance) Substitution Type=1(Delete.)
applied_resource_units			
applied_resource_value			
uom_code			
basis_type			
activity_id			
autocharge_type			
standard_rate_flag			
start_date	required	required	
completion_date	required	required	
schedule_seg_num			
substitute_group_num			
replacement_group_num			
setup_id			
batch_id			
transaction_date	required	required	required
last_update_date	required	required	required
last_update_by	optional	optional	optional

Column Name	Load Type=7(Resource Instance) Substitution Type=4(Recom.)	Load Type=7(Resource Instance) Substitution Type=2(Add.)	Load Type=7(Resource Instance) Substitution Type=1(Delete.)
last_update_by_name	optional	optional	optional
creation_date	required	required	required
created_by	required	required	required
created_by_name	required	required	required
last_update_login	optional	optional	optional

For Loading a Resource Usage

The following table describe column validation for WSM_LOT_JOB_DTL_INTERFACE table for loading a resource usage:

Column Name	Load Type=4(Resource Usage) Substitution Type=4(Recom.)	Load Type=4(Resource Usage) Substitution Type=2(Add.)	Load Type=4(Resource usage) Substitution Type=1(Delete.)
record_id	required	required	required
interface_id			
group_id	required	required	required
parent_header_id	optional	optional	optional
load_type	4 (required)	4 (required)	4 (required)
substitution_type	4 (required)	2 (required)	1 (required)
process_phase			
process_status	1 (required)	1 (required)	1 (required)

Column Name	Load Type=4(Resource Usage) Substitution Type=4(Recom.)	Load Type=4(Resource Usage) Substitution Type=2(Add.)	Load Type=4(Resource usage) Substitution Type=1(Delete.)
job_name	optional	optional	optional
wip_entity_id	optional	optional	optional
organization_id	optional	optional	optional
organization_code	optional	optional	optional
department_id	optional	optional	optional
department_code	optional	optional	optional
description			
job_op_seq_num	required (current op)	required (current op)	required (current op)
routing_op_seq_num	required (future op)	required (future op)	required (future op)
next_routing_op_seq_num	required	required	required
resource_seq_num	required	required	required
resource_id_old			
resource_id_new	optional	optional	
resource_code_old			
resource_code_new	optional	optional	
usage_rate_or_amount			
scheduled_flag			
assigned_units	required	required	

Column Name	Load Type=4(Resource Usage) Substitution Type=4(Recom.)	Load Type=4(Resource Usage) Substitution Type=2(Add.)	Load Type=4(Resource usage) Substitution Type=1(Delete.)
applied_resource_units			
applied_resource_value			
uom_code			
basis_type			
activity_id			
autocharge_type			
standard_rate_flag			
start_date	required	required	
completion_date	required	required	
schedule_seg_num			
substitute_group_num			
replacement_group_num			
setup_id			
batch_id			
transaction_date	required	required	required
last_update_date	required	required	required
last_update_by	optional	optional	optional

Column Name	Load Type=4(Resource Usage) Substitution Type=4(Recom.)	Load Type=4(Resource Usage) Substitution Type=2(Add.)	Load Type=4(Resource usage) Substitution Type=1(Delete.)
last_update_by_name	optional	optional	optional
creation_date	required	required	required
created_by	required	required	required
created_by_name	required	required	required
last_update_login	optional	optional	optional

For Loading a Component

The following table describes column validation for WSM_LOT_JOB_DTL_INTERFACE table for loading a component:

Column Name	Load Type=2(Component) Substitution Type=4(Recom.)	Load Type=2(Component) Substitution Type=3(Change)
record_id	required	required
interface_id		
group_id	required	required
parent_header_id	optional	optional
load_type	2 (required)	2 (required)
substitution_type	4 (required)	3 (required)
process_phase		
process_status	1 (required)	1 (required)
job_name	optional	optional

Column Name	Load Type=2(Component) Substitution Type=4(Recom.)	Load Type=2(Component) Substitution Type=3(Change)
wip_entity_id	optional	optional
organization_id	optional	optional
organization_code	optional	optional
department_id	optional	optional
department_code	optional	optional
description		
job_op_seq_num	required (current op)	required (current op)
routing_op_seq_num	required (future op)	required (future op)
next_routing_op_seq_num	null	null
inventory_item_id_old		
inventory_item_id_new	required	required
inventory_item_old		
inventory_item_new	optional	optional
primary_item_id	required	required
primary_item	optional	optional
src_phantom_item_id	required	required
src_phantom_item	optional	optional
component_seq_id	required	required
quantity_per_assembly		

Column Name	Load Type=2(Component) Substitution Type=4(Recom.)	Load Type=2(Component) Substitution Type=3(Change)
wip_supply_type		
date_required	optional	optional
required_quantity	optional	optional
quantity_issued		
supply_subinventory	optional	optional
supply_locator_id	optional	optional
mrp_net_flag		
mps_required_quantity		
mps_date_required		
transaction_date	required	required
last_update_date	required	required
last_update_by	optional	optional
last_update_by_name	optional	optional
creation_date	required	required
created_by	required	required
created_by_name	required	required
last_update_login	optional	optional

WSM_SERIAL_TXN_INTERFACE Table

Column	Type	Required	Derived	Optional	Permitted Values
HEADER_ID	NUMBER	Y			Header ID of the transaction record in WSM_LOT_JOB_INTERFACE
TRANSACTION_TYPE_ID	NUMBER	Y			Transaction type id 1 – Import Lot based Jobs
SERIAL_NUMBER	VARCHAR2			Y	Serial Number
GENERATE_SERIAL_NUMBER	NUMBER			Y	1 - To generate serial numbers qty provided by generate for qty. If this column is given, then values in SERIAL_NUMBER and ACTION_FLAG are ignored.

Column	Type	Required	Derived	Optional	Permitted Values
GENERATE_FOR_QTY	NUMBER			Y	Number of serials to be generated and associated. This column value is considered only when GENERATE_SERIAL_NUMBER column is populated with 1.
ACTION_FLAG	NUMBER			Y	Action to be performed on the specified serial Number mentioned in SERIAL_NUMBER column: 1 - Add, 2 – De link, 3 - Update
SERIAL_ATTRIBUTE_CATEGORY, CATTRIBUTE1-20, DATTRIBUTE1-10, NATTRIBUTE1-10 (Unnamed columns – Serial attributes)				Y	

Column	Type	Required	Derived	Optional	Permitted Values
STATUS_ID TIME_SINCE_NEW CYCLES_SINCE_NEW TIME_SINCE_OVERHAUL CYCLES_SINCE_OVERHAUL TIME_SINCE_REPAIR CYCLES_SINCE_REPAIR TIME_SINCE_VISIT CYCLES_SINCE_VISIT TIME_SINCE_MARK CYCLES_SINCE_MARK NUMBER_OF_REPAIRS (Named columns – Serial attributes)	NUMBER			Y	(Serial Attributes Columns)
ATTRIBUTE_CATEGORY, ATTRIBUTE1-15				Y	(Serial Descriptive flex field columns)

Mode-2 Job Creation through Interface

Following are the major design features of Mode 2 job creation through the interfaces:

1. You can create a lot based job of any name from a source lot as long as the name is unique.

2. Jobs can be created for any quantity as long as the quantity required to be issued from the inventory lot is less than or equal to the available quantity in the source

lot.

3. Thus job quantity will be checked against the actual available quantity determined using the quantity tree.

4. A parameter 'group_id' is introduced to WSM_LOT_JOB_INTERFACES, when user launches 'Import Lot Based Job' with a group_id. Rows with the given group_id will be picked up by the concurrent program.

5. For creating a mode 1 job, insert a row into WSM_LOT_JOB_INTERFACE. For creating a mode 2 job, additionally insert a row into WSM_STARTING_LOTS_INTERFACE table. For a mode 2 job, the HEADER_ID in WSLI should be equal to the source line id in WLJI. Oracle Shop Floor Management, from the Run Requests menu, select Import Lot Based Jobs.

6. Mode 2 Job creation is currently not supported for a serial control component inventory lot.

WSM_STARTING_LOTS_INTERFACE Table

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number	x			
lot_number	varchar2(30)	x			
inventory_item_id	number	x			
organization_id	number	x			
quantity	number	x			This quantiy is ignored

Column	Type	Required	Derived	Optional	Permitted Values
component_issue_quantity	number		x		If not entered, this quantity will be derived from job quantity and quantity per assembly defined in BOM; Otherwise, it will be honored
subinventory_code	varchar2(10)	x			
last_update_date	date	x			
last_update_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number			x	

Lot Move Transactions Concurrent Program

You can load Move Transaction information into the Open Move Transaction Interface table from a variety of sources, including external data collection devices such as bar code readers, automated test equipment or other manufacturing execution systems.

Use the interface to load these transactions into the appropriate tables. All transactions are validated. Invalid transactions are marked so that you can correct and resubmit them.

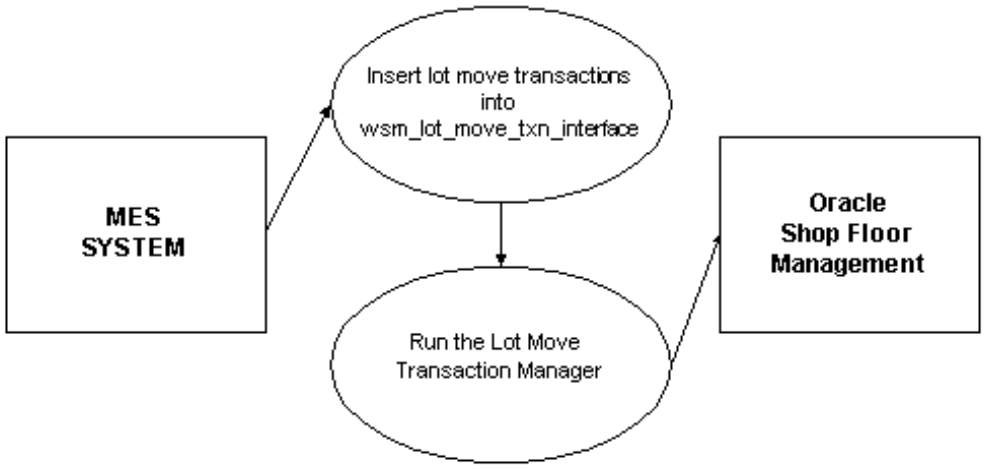
The open interface enables you to perform many of the functions possible from the Move Transactions form. For example, you can:

- Move jobs between operations and intraoperation steps
- Move some quantity and scrap remaining quantity for jobs between operations and intraoperation steps. Scrap can be done at either the start or destination operation
- Jump to any standard operation defined inside or outside the network routing
- Jump as described above and scrap some quantity
- Scrap certain quantities of a job
- Scrap some quantity and complete remaining quantity into inventory
- Return assemblies into WIP
- Return assemblies into WIP and undo any scrap done during completion
- Undo the last move, jump, or scrap transaction

Lot move transactions are processed in parallel. The Move concurrent transaction manager (WSCMTM) program picks up a set of records from wsm_lot_move_txn_interface, assigns a group id to them and spawns any necessary workers (WSCMTI).

The following diagram describes the basic flow of the Lot Move Transactions interface. The lot move transactions in MES are inserted into the WSM_LOT_MOVE_TXN_INTERFACE table by running a script. The Move Transactions worker validates the lot move transactions to be interfaced. The lot move transactions from MES are then imported into Oracle Applications.

Basic Flow of the Lot Move Transactions Concurrent Program



Serial Support in Lot Move Interface

The following features are supported for serial controlled assembly of a lot based job through Lot Move Interface:

- Associate existing serial numbers for a lot based job.
- Generate and associate new serials to a lot based job.
- Disassociate serial numbers from a lot based job.
- Scrap serial numbers while performing scrap transaction for a lot based job.
- Specify serial attributes information for the serial numbers.
- Initiate serial tracking for a lot based job.

The following feature is not supported through Lot Move Interface:

- Backflushing serials for serial controlled components.

The Lot Move Interface program processes Serial Number details by linking the data specified in WSM_LOT_MOVE_TXN_INTERFACE and WSM_SERIAL_TXN_INTERFACE tables.

Lot Move Transactions Concurrent Program Features

The interface requires WSM_LOT_MOVE_TXN_INTERFACE (WLMTI) and WIP_MOVE_TXN_INTERFACE (WMTI) tables.

The requisite data is to be inserted into the WLMTI table only.

Functional Overview

You must write the load program that inserts a single row for each move transaction into the WSM_LOT_MOVE_TXN_INTERFACE table. The system uses this information to calculate completion cost. The Move Transaction Manager selects the pending move transactions for processing and spawns any necessary workers.

The Move Transaction Worker calls the WSM Transaction Validation Engine program, which validates the row, derives or defaults any additional columns, and inserts errors into the WSM_INTERFACE_ERRORS table.

The Backflush Setup program determines and initiates related operation pull backflushes. Next, the Move Transaction Processor performs the actual move transaction. It writes it to history, initiates related resource and overhead transactions and requisitions for outside resources (for outside processing only), and deletes successfully processed transaction rows from the WSM_LOT_MOVE_TXN_INTERFACE table based on the profile 'WSM: Open Interface successful transactions archive days'

Following are the major design features of this interface:

1. Insert rows into the WSM_LOT_MOVE_TXN_INTERFACE (WLMTI) table. HEADER_ID is populated using the sequence given in the table requirements section below.

2. The transaction manager picks up a transaction for processing according to the following criteria:
   - The group_id must be NULL
   - A valid wip_entity_id or wip_entity_name
   - The transaction date is no later than the current date
   - A transaction from the same job is not currently running in another move worker.

3. The manager assigns a set of move transactions to one or more workers in the following order: transaction_date, organization_id, wip_entity_id, and processing_order. The number of transactions that are assigned to a worker are determined by the worker rows that you set up for Lot Move transactions. To make this change, select Interface Managers under Run Requests in Oracle Shop Floor Management.

4. Although many transactions can be processed simultaneously, only one worker at a time can process transactions that belong to the same job.

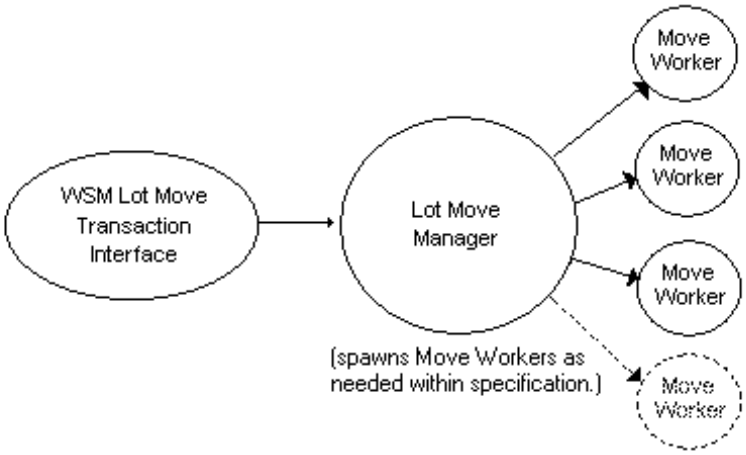
5. In WSM_SERIALTXN_INTERFACE table populate columns GENERATE_SERIAL_NUMBER and GENERATE_FOR_QTY with appropriate values to generate and associate serials. While transacting with existing serials, populate columns SERIAL_NUMBER and ACTION_FLAG with appropriate values to associate/update/disassociate/scrap the serials.

6. The user can also initiate serial tracking for a non-serial tracked job during a move transaction by specifying 1 for the column START_SERIAL_TRACK in the interface table WSM_LOT_MOVE_TXN_INTERFACE.

7. For information related to serial attributes, please refer **Support for Serial Attributes through Interfaces**, page 10-4.

The following diagram describes the basic flow of the Lot Move Transactions Manager. The manager picks up pending records in wsm_lot_move_txn_interface and processes them in parallel, starting workers as necessary.

Lot Move Transaction Manager



Validation of Import Lot Based Move Transactions

The following describes validation of import lot-based move transactions.

Standard Validation

Oracle Shop Floor Management validates all required columns in the Import Lot Based Move Transactions Concurrent Program. For specific information on the data implied by these columns, see Oracle Shop Floor Management Technical Reference Manual or eTRM for details.

Error Handling

If any validation fails, the Concurrent Program will return error status to the calling module. The Import Lot Based Move Transactions Concurrent Program processes the rows and reports the following values for every record.

1. Whenever any transaction errors out, the error column contains the error message.

2. Errors and warnings, if any, are also written into the WSM_LOT_MOVE_TXN_INTERFACE table and the WSM_INTERFACE_ERRORS table.

3. To resubmit the errored transactions for processing, first rectify the error and then do the following:

   - Change the PROCESS_STATUS of the required transactions from 3 (ERROR) to 1 (PENDING).

   - Set the group_id to NULL.

4. The interface makes the following validation checks:

   - The details of the job are correct. The details include the job name, qty, uom's, current status in terms of operation sequence number, operation code, department, etc.

   - The details of the operation to which the move is to be done are valid. These include TO_OPERATION_SEQ_NUM, TO_OPERATION_CODE, TO_DEPARTMENT_ID, and TO_INTRAOPERATION_STEP_TYPE.

   - For return transactions, the job must have been completed.

5. In case of any error in serial processing,then the parent move transaction is also errored out and the error message is written into WSM_INTERFACE_ERRORS table.

Error Handling Table

Condition	PROCESS_STATIS	ERROR_MESSAGE
Success	4	null
Failure	3	actual error message
Running	2	null

See Also

Oracle Shop Floor Management Technical Reference Manual

Setting Up the Move Transaction Interface

You must perform all the Oracle Bills of Material, Oracle Work in Process and Oracle Shop Floor Management setup activities required for move transactions. In addition, you must launch the Lot Move Transactions Manager to import from external sources.

Launching the Move Transaction Processor

Launch the Lot Move Transaction Manager in the Interface Managers window by selecting Interface Managers under Run Requests in Oracle Shop Floor Management. You can specify the resubmit interval and number of transactions processed by each worker during each interval when you launch the Lot Move Transaction Manager. After polling the WSM_LOT_MOVE_TXN_INTERFACE table for eligible rows, the Lot Move Transaction Manager creates the necessary number of Move Transaction Workers to process the load.

The use of multiple transaction workers enables parallel processing of transactions that can be especially helpful when importing a large batch of transactions through the Move Transaction Interface. For more information, see: *Transaction Managers, Oracle Inventory User's Guide.*

Inserting Records into the WSM_LOT_MOVE_TXN_INTERFACE Table

You must insert your Move, Move Complete and Move Return transactions into the WSM_LOT_MOVE_TXN_INTERFACE table. The system validates each transaction row, derives any additional data as necessary, then processes each transaction.

The following tables describe the WSM_LOT_MOVE_TXN_INTERFACE table:

Number Types and Descriptions

Number Type	Transaction Type Description
1	Move Transaction (defined by TRANSACTION_TYPE = 1 and valid from and to operation information)
2	Move Completion (defined by TRANSACTION_TYPE = 2)
3	Move Return (defined by TRANSACTION_TYPE = 3)
4	Move Undo (defined by TRANSACTION_TYPE = 4)

Forward Moves and Completions WSM_LOT_MOVE_TXN_INTERFACE (WLMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number	=wsm_lot_move_txn_interface_s			
transaction_id	number		wip_interface_s		
last_update_date	date	x			
last_updated_by	number	x			
last_updated_by_name	varchar2(100)		x	x	
creation_date	date	x			
created_by	number	x			
created_by_name	varchar2(100)		x	x	
last_update_login	number				
request_id	number				
program_application_id	number				
program_id	number				
program_update_date	date				
group_id	number		wip_interface_s		
source_code	varchar2(30)				

Column	Type	Required	Derived	Optional	Permitted Values
source_line_id	number				
status	number	=1 (PENDING)			
transaction_type	number	=1 (MOVE) / 2 (COMPLETION)			
organization_id	number	x			
organization_code	varchar2(3)		x	x	
wip_entity_id	number		x	x	
wip_entity_name	varchar2(240)	x			
entity_type	number	=5			
primary_item_id	number		x	x	
line_id	number				
line_code	varchar2(10)				
repetitive_schedule_id	number				
transaction_date	date	x			
acct_period_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
fm_operation_seq_num	number	x			
fm_operation_code	varchar2(4)	x			
fm_department_id	number	x			
fm_department_code	varchar2(10)		x		
fm_intraoperation_step_type	number	x			
to_operation_seq_num	number	x			
to_operation_code	varchar2(4)	x			
to_department_id	number	x			
to_department_code	varchar2(10)		x		
to_intraoperation_step_type	number	x			
transaction_quantity	number	x			
transaction_uom	varchar2(3)	x			
primary_quantity	number				

Column	Type	Required	Derived	Optional	Permitted Values
primary_uom	varchar2(3)	x			
scrap_account_id	number	for scrap txns			
reason_id	number				
reason_name	varchar2(30)				
reference	varchar2(240)				
attribute_category	varchar2(30)				
attribute1 - 15	varchar2(150)				
qa_collection_id	number				
kanban_card_id	number				
overcompletion_transaction_qty	number				
overcompletion_primary_qty	number				
overcompletion_transaction_id	number				
error	varchar2(240)				
jump_flag	varchar2(1)	for Jumps			
processing_order	number				

Column	Type	Required	Derived	Optional	Permitted Values
scrap_quantity	number			x	
primary_scrap_quantity	number				
scrap_at_operation_flag	number	x(for interoperation move and scrap)			
start_serial_track	number			Y	1 – to initiate serial tracking for a lot based job

UNDO MOVES WSM_LOT_MOVE_TXN_INTERFACE (WLMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number	=wsm_lot_move_txn_interface_s			
transaction_id	number		wip_transaction_s		
last_update_date	date	x			
last_updated_by	number	x			
last_updated_by_name	varchar2(100)		x	x	
creation_date	date	x			
created_by	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
created_by_name	varchar2(100)		x	x	
last_update_login	number				
request_id	number				
program_application_id	number				
program_id	number				
program_update_date	date				
group_id	number		wip_interface_s		
source_code	varchar2(30)				
source_line_id	number				
status	number	=1 (PENDING)			
transaction_type	number	=4 (UNDO)			
organization_id	number	x			
organization_code	varchar2(3)		x	x	
wip_entity_id	number		x	x	
wip_entity_name	varchar2(240)	x			

Column	Type	Required	Derived	Optional	Permitted Values
entity_type	number	=5			
primary_item_id	number		x	x	
line_id	number				
line_code	varchar2(10)				
repetitive_schedule_id	number				
transaction_date	date	x			
acct_period_id	number				
fm_operation_seq_num	number		x	x	
fm_operation_code	varchar2(4)		x	x	
fm_department_id	number		x	x	
fm_department_code	varchar2(10)		x		
fm_intraoperation_step_type	number		x	x	
to_operation_seq_num	number		x	x	
to_operation_code	varchar2(4)		x	x	

Column	Type	Required	Derived	Optional	Permitted Values
to_department_id	number		x	x	
to_department_code	varchar2(10)		x		
to_intraoperation_step_type	number		x	x	
transaction_quantity	number		x	x	
transaction_uom	varchar2(3)	x			
primary_quantity	number				
primary_uom	varchar2(3)	x			
scrap_account_id	number		for scrap txns	for scrap txns	
reason_id	number				
reason_name	varchar2(30)				
reference	varchar2(240)				
attribute_category	varchar2(30)				
attribute1 - 15	varchar2(150)				
qa_collection_id	number				
kanban_card_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
overcompletion_transaction_qty	number				
overcompletion_primary_qty	number				
overcompletion_transaction_id	number				
error	varchar2(240)				
jump_flag	varchar2(1)				
processing_order	number				
scrap_quantity	number	x(for scrap only txns either scrap_quantity or transaction_quantity should be populated)	x(for move and scrap txns)	x(for move and scrap txns)	
primary_scrap_quantity	number				
scrap_at_operation_flag	number		x	x	1 or 2

Returns WSM_LOT_MOVE_TXN_INTERFACE (WLMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number	=wsm_lot_move_txn_interface_s			
transaction_id	number		wip_transaction_s		
last_update_date	date	x			
last_updated_by	number	x			
last_updated_by_name	varchar2(100)		x	x	
creation_date	date	x			
created_by	number	x			
created_by_name	varchar2(100)		x	x	
last_update_login	number				
request_id	number				
program_application_id	number				
program_id	number				
program_update_date	date				
group_id	number		wip_interface_s		

Column	Type	Required	Derived	Optional	Permitted Values
source_code	varchar2(30)				
source_line_id	number				
status	number	=1 (PENDING)			
transaction_type	number	=3 (RETURN)			
organization_id	number	x			
organization_code	varchar2(3)		x	x	
wip_entity_id	number		x	x	
wip_entity_name	varchar2(240)	x			
entity_type	number	=5			
primary_item_id	number		x	x	
line_id	number				
line_code	varchar2(10)				
repetitive_schedule_id	number				
transaction_date	date	x			
acct_period_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
fm_operation_seq_num	number		x	x	
fm_operation_code	varchar2(4)		x	x	
fm_department_id	number		x	x	
fm_department_code	varchar2(10)		x		
fm_intraoperation_step_type	number		x	x	
to_operation_seq_num	number		x	x	
to_operation_code	varchar2(4)		x	x	
to_department_id	number		x	x	
to_department_code	varchar2(10)		x		
to_intraoperation_step_type	number		x	x	
transaction_quantity	number		x	x	
transaction_uom	varchar2(3)	x			
primary_quantity	number				

Column	Type	Required	Derived	Optional	Permitted Values
primary_uom	varchar2(3)	x			
scrap_account_id	number				
reason_id	number				
reason_name	varchar2(30)				
reference	varchar2(240)				
attribute_category	varchar2(30)				
attribute1 - 15	varchar2(150)				
qa_collection_id	number				
kanban_card_id	number				
overcompletion_transaction_qty	number				
overcompletion_primary_qty	number				
overcompletion_transaction_id	number				
error	varchar2(240)				
jump_flag	varchar2(1)				
processing_order	number				

Column	Type	Required	Derived	Optional	Permitted Values
scrap_quantity	number		x	x	
primary_scrap_quantity	number				
scrap_at_operation_flag	number		x	x	1 or 2

WSM_SERIAL_TXN_INTERFACE Table

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number				Header ID of the transaction record in (WSM_LOT_MOVE_TXN_INTERFACE)
transaction_type_id	number				Transaction type id 2 - Lot based Job Move transaction
serial_number	varchar2				Serial Number

Column	Type	Required	Derived	Optional	Permitted Values
generate_serial_number	number				1 - To generate serial numbers qty provided by generate for qty. If this Column is given, then values in SERIAL_NUMBER and ACTION_FLAG are ignored.
generate_for_qty	number				Number of serials to be generated and associated. This column value is considered only when generate_serial_number column is populated with 1.
action_flag	number				Action to be performed on the specified serial number mentioned in serial_number column. 1 - Add, 2 -De link, 3 - Update, 5 -Move and 6 - Scrap

Column	Type	Required	Derived	Optional	Permitted Values
serial_attribute_category, cattribute1-20, dattribute1-10, nattribute1-10 (Unnamed columns – serial attributes)					(Serial Attribute Columns)
status_id time_since_new cycles_since_new time_since_overhaul cycles_since_overhaul time_since_repair cycles_since_repair time_since_visit cycles_since_visit time_since_mark cycles_since_mark number_of_repairs (named columns – serial attributes)	number				(serial attributes columns)
attribute_category, attribute1-15					(serial descriptive flex field columns)

You must include data in each of the required columns. Overall, very few columns are required because the system derives or defaults many column values and/or allows these column values to be optional.

Columns are derived using foreign key relationships within Oracle Manufacturing. The following derived columns are control columns that the Move Transaction Manager uses to provide closed loop transaction processing control and relational integrity throughout the interface process:

Control Columns

- ATTRIBUTE1 through ATTRIBUTE15 (Optional): the descriptive flexfield attributes in the columns ATTRIBUTE1 through ATTRIBUTE15 map to ATTRIBUTE1 through ATTRIBUTE15 in WIP_MOVE_TRANSACTIONS.

- FM_INTRAOPERATION_STEP_TYPE (Required): this column is only required when performing Move and Move Completion transactions. It must be an enabled intraoperation step.

- FM_OPERATION_SEQ_NUM (Required): in Move transactions, this column represents the operation from which you are moving the jobs.

  In Move and Completion transactions, this column represents the operation from which you are moving the jobs before they are completed into inventory.

  In Undo Move and Return transactions, you may leave this column and the FM_INTRAOPERATION_STEP_TYPE column blank. If you do not wish to leave these columns blank when performing an Undo Move and Return transaction, you must set the values of these columns to their derived values. FM_OPERATION_SEQ_NUM and FM_INTRAOPERATION_STEP_TYPE must be set to the operation sequence on the routing at which the job currently exists.

- ORGANIZATION_CODE (Derived): this column is derived from the Organization ID. The Organization ID identifies the organization to which the transaction belongs.

- PRIMARY_QUANTITY (Derived): this column is the transaction quantity in the assembly's primary unit of measure, calculated using TRANSACTION_QUANTITY and TRANSACTION_UOM.

- STATUS (Required): this column control describes the transaction state of the row and controls whether rows in the interface table are processed. You should insert a row that you intend to be processed with a value of 1.

  1 Pending

  2 Running

  3 Error

  4 Completed

You should always load 1 (Pending)

- SCRAP_ACCOUNT_ID (Optional): if the TO_INTRAOPERATION_STEP_TYPE is scrap and a scrap account is required, you must insert a SCRAP_ACCOUNT_ID.

- SOURCE_CODE and SOURCE_LINE_ID (Optional): the SOURCE_CODE and SOURCE_LINE_ID columns can be used to identify the source of your Move transactions. For example, if you collect Move transaction information from a bar code reader and a radio frequency device, you could use a different source code to identify each collection method.

- TO_INTRAOPERATION_STEP_TYPE (Required): If you are undoing a move or returning an assembly from inventory back to WIP, you cannot specify the To Move intraoperation step. If you specify the Scrap intraoperation step, you must insert a SCRAP_ACCOUNT_ID if the WIP Require Scrap Account parameter is set.

- TO_OPERATION_SEQ_NUM (Required): in Move transactions, this column represents the operation step into which you are moving the job. If you are undoing a move or returning an assembly from inventory to WIP leave this column and the TO_INTRAOPERATION_STEP_TYPE columns blank, both columns are derived.

  For forward move transactions and jumps within the current routing, the TO_OPERATION_SEQ_NUM should be populated with the OPERATION_SEQUENCE_NUMBER defined in the network routing.

  For undo transactions, this column need not be populated.

  For jumps outside the network routing this column must be left blank.

  In Return transactions, this column represents the operation that the assemblies are being returned to from inventory.

  JUMP_FLAG: Jump is a move transaction in which the operation to which the job is to be moved is not the immediately next operation on the network routing attached to the job. Jumps can be made either within the same network routing or outside of the network routing. Whenever jumps are made, the jump flag needs to be set to Y.

- TRANSACTION_QUANTITY (Required): enter the transaction quantity in the same unit of measure used in the transaction.

- TRANSACTION_TYPE (Optional): This column indicates the type of Move transaction. The options are:

  - 1 Move

  - 2 Move Completion

  - 3 Move Return

  - 4 Undo

If the transaction type is set to 1 and you are moving into the last operation To Move intraoperation step, the program defaults the transaction type to 2 and performs the completion transaction.

- TRANSACTION_UOM (Required): you can enter the TRANSACTION_QUANTITY in any unit of measure that has conversion rates defined for the item's primary unit of measure. Use this column to specify the transacted unit of measure, even if it is the same as the primary unit of measure.

- WIP_ENTITY_NAME (Required): this column represents the job name used to derive the WIP_ENTITY_ID.

Required Columns

- For normal Move transactions, set TRANSACTION_TYPE to 1 or NULL.

- The FROM and TO OPERATION_SEQ_NUM, OPERATION_CODE, INTRAOPERATION_STEP, and DEPARTMENT_ID must be set by the user.

- For completion transactions, set TRANSACTION_TYPE to 2.

- For return transactions, set TRANSACTION_TYPE to 3. When TRANSACTION_TYPE is 3, the Move transaction processor returns the assemblies from the completion subinventory/locator back into WIP.

- Scrap_at_operation_flag -This is required when performing a move and scrap transaction to indicate where the quantity should be scrapped at the start or the destination operation.

- The column STATUS contains the state of the transaction. You should always load 1 (Pending):
  - 1 Pending
  - 2 Running
  - 3 Error
  - 4 Complete

Derived Data

The WSM Transaction Validation Engine derives columns using foreign key relationships within Oracle Manufacturing. The following derived columns are control columns that the Move Transaction Worker uses to provide closed loop transaction processing control and relational integrity throughout the interface process:

- CREATED_BY_NAME

- GROUP_ID
- LAST_UPDATED_BY_NAME
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE
- REQUEST_ID
- TRANSACTION_ID

You can insert data into certain derived columns. The WSM Transaction Validation Engine will validate your data, but not override it. You can insert data into the following derived columns:

- LINE_ID
- REASON_ID

Optional Columns

- The columns SOURCE_CODE and SOURCE_LINE_ID can be used to identify the source of Move transactions. For example, if you collect Move transaction information from a bar code reader and a radio frequency device, you could use a different source code to identify each collection method.
- The descriptive flexfield attributes in the columns ATTRIBUTE1 through ATTRIBUTE15 map to ATTRIBUTE1 through ATTRIBUTE15 in WIP_MOVE_TRANSACTIONS.
- Scrap_quantity: This column is optional and needs to be populated when performing a move abd scrap transaction. For scrap only transactions either the scrap_quantity or transaction_quantity should be populated.

Import WIP Lot Transactions Concurrent Program

The WIP Lot Transactions interface supports the following types of transactions:

Transaction Types and Descriptions

Transaction Type	Transaction Description
1	Split

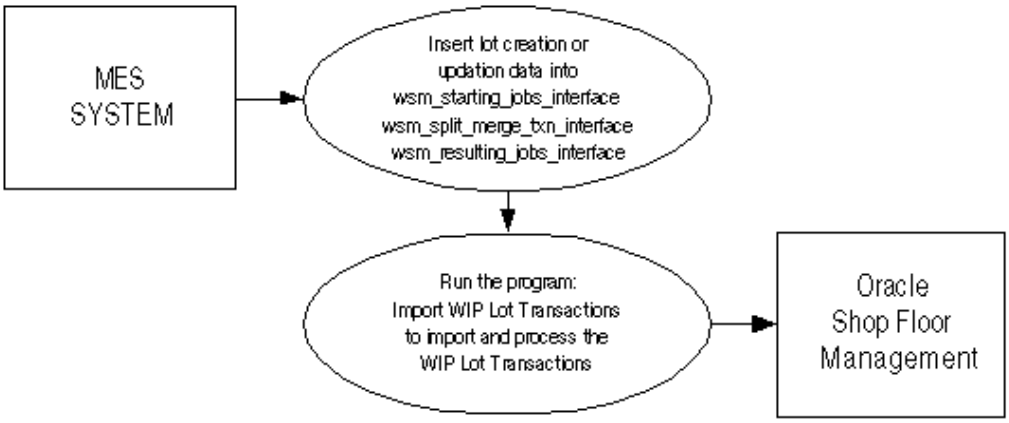
Transaction Type	Transaction Description
2	Merge
3	Update Assembly
4	Bonus
5	Update Routing
6	Update Quantity
7	Update Lot Name

The transactions in this interface can be broadly divided into two categories: transactions and update records. Split, merge and bonus can be classified as transactions, where as updates of assembly, routings, quantity and lot names can be classified as update records. In addition, split transaction now allows update of assembly when the split_has_update_assembly flag is set.

This is a two step process. First, use the concurrent program Import WIP Lot Transactions to validate each transaction record in the interfaces tables, import it into the base tables and then process it. The PROCESS_STATUS of affected transaction records in the base tables will be COMPLETE once these transaction have been successfully imported and processed. Finally, use the concurrent program Cost Manager to cost the record with process_status = complete. After this step, the COSTED status of the records will also be changed to COMPLETE.

The following diagram illustrates the basic flow of the WIP Lot Transactions Concurrent Program for transactions. The WIP split, merge, update and bonus lot data in MES is inserted into Oracle Shop Floor Management interface tables by running interface loader scripts from the external system. This import program validates the WIP split, merge, update and bonus lot data to be interfaced. The WIP split, merge, update and bonus lot transactions in MES are then imported into Oracle Applications.

Basic Flow of the WIP Lot Transactions Concurrent Program for Transactions



Serial Support in WIP Lot Transactions Interface

The following features are supported for serial controlled assembly of a lot based job through WIP Lot Transactions Interface:

- Associate serial numbers from parent job to resulting job during split transaction.
- Generate and associate new serials to a lot based job during update quantity transaction.
- Associate existing serials to a lot based job during update quantity transaction.
- Specify serial attributes information for the serial numbers.

The following feature is not supported through WIP Lot Transactions Interface:

- For Bonus transactions, serials are not supported. As a work around users can create bonus job and associate serials to the job using Import lot jobs concurrent program.

The WIP Lot Transactions Interface program processes Serial Number details by linking the data specified in WSM_SPLIT_MERGE_TXN_INTERFACE and WSM_SERIALTXN_INTERFACE tables.

Functional Overview

In Oracle Shop Floor Management, the run Requests menu, select Import WIP Lot Transactions.

To run from the command line, enter the following command:

WSMPLOAD.load (errbuf OUT VARCHAR2,

retcode OUT NUMBER,

p_copy_qa IN VARCHAR2,

p_group_id IN NUMBER NULL);

or

WSMPLOAD.load (errbuf OUT VARCHAR2,

retcode OUT NUMBER,

p_copy_qa IN VARCHAR2,

p_group_id IN NUMBER NULL,

p_copy_flag IN NUMBER NULL);

Following are the major design features of this interface:

1. Insert rows into WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI), WSM_STARTING_JOBS_INTERFACE (WSJI), and WSM_RESULTING_JOBS_INTERFACE (WRJI) tables. Transactions in WSMTI are joined with the transactions in WSJI and WRJI by the HEADER_ID column in these tables.

2. Group transactions in WSMTI by using the same GROUP_ID. He/she may or may not enter a GROUP_ID.

3. When a concurrent request is launched for WIP Lot Transaction Interface, you can specify a GROUP_ID. If a GROUP_ID is specified, only the PENDING transactions with the given GROUP_ID will be processed. If no GROUP_ID is specified, all the PENDING transactions are processed group by group. In this case, a unique GROUP_ID is assigned for each PENDING transaction which has a NULL GROUP_ID, by the order of TRANSACTION_DATE.

4. Within a group, transactions are processed according to TRANSACTION_DATE.

5. In WSM_SERIALTXN_INTERFACE table, populate columns GENERATE_SERIAL_NUMBER and GENERATE_FOR_QTY with appropriate values to generate and associate serials. While transacting with existing serials, populate columns SERIAL_NUMBER and ACTION_FLAG with appropriate values to associate/update/disassociate the serials.

6. For information related to serial attributes, please refer **Support for Serial Attributes through Interfaces**, page 10-4.

Errors and Validations:

1. Whenever any one row errors out within a given group, the PROCESS_STATUS of the entire group is set to ERROR. But only the transactions that actually errored out will have the ERROR_MESSAGE column containing the error message.

2. Preferably, independent transactions should be grouped together. No interdependent transactions should exist within a group. Ideally, the GROUP_ID should be left blank so that each transaction is assigned a unique GROUP_ID according to TRANSACTION_DATE and the transactions are processed row-by-row as if they were a group with a single transaction in it.

3. If GROUP_ID is not specified when a concurrent request is launched and any one group errors out, the status of the concurrent program is set to WARNING. On the other hand, if a GROUP_ID is specified when a concurrent request is launched and the group errors out, the status of the concurrent program is set to ERROR.

4. Errors and warnings, if any, will be written into the WSM_INTERFACE_ERRORS table. Appropriate messages will be given in the parent and child interface tables to refer to WSM_INTERFACE_ERRORS table when there is any error.

5. To resubmit the errored transactions for processing, you must correct the error and change the PROCESS_STATUS of the required transactions from 3 (ERROR) to 1 (PENDING).

6. The interface makes all the validations that get done when going though the front-end.

7. In case of any error in serial processing, then the parent WIP lot transaction is also errored out and the error message is written into WSM_INTERFACE_ERRORS table.

Setting Up the Import WIP Lot Transactions Concurrent Program

The following describes setting up the import WIP lot transactions concurrent program.

Parameter Descriptions

The following chart describes all parameters used by the public WIP LOT Transaction Concurrent Program(WSMPLOAD.load). All of the inbound and outbound parameters are listed.

Parameter Descriptions

Parameter	Usage	Type	Description
errbuf	out	VARCHAR2	Contains the error message
retcode	out	NUMBER	Contains the error code

Parameter	Usage	Type	Description
p_copy_qa	in	VARCHAR2	Currently not used
p_group_id	in	NUMBER	Optional. Identifier to process a group of transaction
p_copy_flag	in	NUMBER	Optional Used when value of site level profile "WSM: Create Job Level BOM and Routing Copies" is YES. 1 - Implies "make copies after processing each transaction 2 - Implies "make copies at end, afetr processing all transactions". Used to improve performance

The following chart describes all parameters used by the Import WIP LOT Transaction Concurrent Program Request.

Parameter Descriptions

Parameter	Possible values	Description
groupID	Identifier (number) of a group of transactions to be processed	Optional. Identifier to process a group of transactions

Parameter	Possible values	Description
make copies after each transaction	Yes/No	Optional. Used when value of site level profile "WSM: Create Job Level BOM and Routing Copies" is YES. YES -Implies "make copies after processing each transaction. NO - Implies "make copies at end, after processing all transactions", used to improve performance.

Validation of Import WIP Lot Transactions Program

The following describes validation of the import WIP lot transactions program.

Standard Validation

Oracle Shop Floor Management validates all required columns in the Import WIP Lot Processor Transactions Concurrent Program. For specific information on the data implied by these columns, see your Oracle Shop Floor Management Technical Reference Manual for details.

Error Handling

If any validation fails, the Concurrent Program will return error status to the calling module. The Import WIP Lot Processor Transactions Concurrent Program processes the rows and reports the following values for every record.

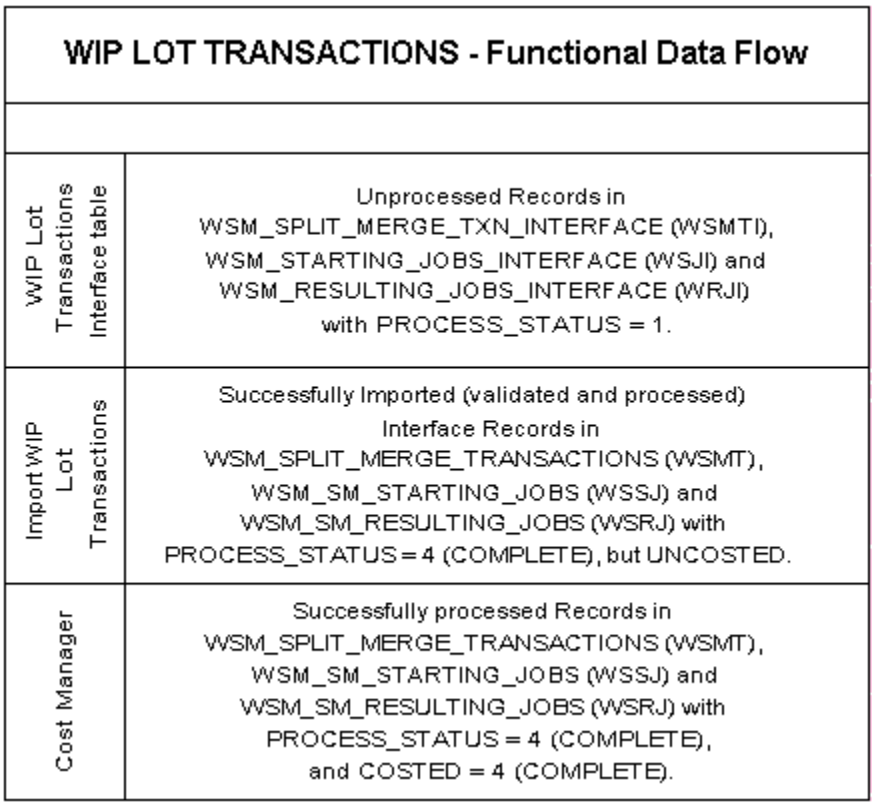
Error Handling Table

Condition	PROCESS_STATIS	ERROR_MESSAGE
Success	4	null
Failure	3	actual error message
Running	2	null

See Also

Oracle Shop Floor Management Technical Reference Manual

The following diagram illustrates the same process for updating records.



Important Columns in WSM_SPLIT_MERGE_TXN_INTERFACE, WSM_STARTING_JOBS_INTERFACE, and WSM_RESULTING_JOBS_INTERFACE Tables

The interface requires WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI), WSM_STARTING_JOBS_INTERFACE (WSJI), and WSM_RESULTING_JOBS_INTERFACE (WRJI) tables.

The system validates each transaction row, derives any additional data as necessary, then processes each transaction. Some columns are not used by the program and marked as ignored.

The following three tables describe the important columns in the WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI), WSM_STARTING_JOBS_INTERFACE (WSJI,) and WSM_RESULTING_JOBS_INTERFACE (WRJI) tables.

IMPORTANT COLUMNS WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Description
group_id	User populated / Program generated if NULL, using sequence wsm_sm_txn_int_group_s
header_id	User populated - using sequence wsm_sm_txn_interface_s
transaction_id	Program generated using sequence wsm_split_merge_transactions_s
transaction_type_id	Type of transaction: 1 - Split 2 - Merge 3 - Update Assembly 4 - Bonus 5 - Update Routing 6 - Update Quantity 7 - Update Lot Name
process_status	1 - PENDING

IMPORTANT COLUMNS WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Description
group_id	User populated / System generated if NULL. Value same as that in WSMTI table
header_id	User populated - using sequence wsm_sm_txn_interface_s. Matches the value for the txn entered in WSMTI table
process_status	1 - PENDING

IMPORTANT COLUMNS WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Description
group_id	User populated / System generated if NULL. Value same as that in WSMTI table
header_id	User populated - using sequence wsm_sm_txn_interface_s. Matches the value for the txn entered in WSMTI table
process_status	1 - PENDING

Inserting Records into the WSM_SPLIT_MERGE_TXN_INTERFACE, WSM_STARTING_JOBS_INTERFACE, and WSM_RESULTING_JOBS_INTERFACE Tables

The following tables describe WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI), WSM_STARTING_JOBS_INTERFACE (WSJI), and WSM_RESULTING_JOBS_INTERFACE (WRJI) tables for each of the seven transaction types.

Bonus Transaction:

BONUS WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			
transaction_type_id	number(15)	=4			
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				

Column	Type	Required	Derived	Optional	Permitted Values
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	=1 (PENDING)			
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System generated
program_application_id	number(15)				System generated

Column	Type	Required	Derived	Optional	Permitted Values
program_id	number(15)				System generated
program_update_date	date				System generated
xml_document_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten

Note: No records are required for insertion to WSJI table for bonus transactions.

BONUS WSM_RESULTIING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL		

Column	Type	Required	Derived	Optional	Permitted Values
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				
description	varchar2(240)				
primary_item_id	number	x			
class_code	varchar2(10)		x	x	If class_code can not be derived using defaults, then the record will error out.
start_quantity	number	x			
bom_reference_id	number				
routing_reference_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
common_bom_sequence_id	number		x	x	Derived using organization_id, primary_item_id and alternate_bom_designator.
common_routing_sequence_id	number		x	x	Derived using organization_id, primary_item_id and alternated_routing_designator. If NO Network routing exists for this condition, then error out.
routing_revision	varchar2(3)		x	x	
routing_revision_date	date		x	x	
bom_revision	varchar2(3)		x	x	
bom_revision_date	date		x	x	
alternate_bom_designator	varchar2(10)		x	x	NULL value is PRIMARY BOM.

Column	Type	Required	Derived	Optional	Permitted Values
alternate_routing_designator	varchar2(10)		x	x	NULL value is PRIMARY Routing.
completion_subinventory	varchar2(10)		x		
starting_operation_code	varchar2(4)		x (if op_seq_num given)	x	If starting operation is a standard operation.
starting_operation_seq_num	number		x (if op_code given)	x	Starting Operation Sequence Number in the PRIMARY PATH of the network routing. This can be a Non-standard operation also.
starting_std_op_id	number		x (if op_code / op_seq_num given)		
starting_intraoperation_step	number		x = 1 (QUEUE)	x	BONUS Transactions are supported only at QUEUE Intra-operation step.
scheduled_start_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
scheduled_completion_date	date	x			
starting_wip_entity_rename	varchar2(240)				
forward_op_option	number		x = 4		
bonus_acct_id	number(15)	x			Should be a valid account id from the organization's Set of Books.
demand_class	varchar2(30)				
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				

Column	Type	Required	Derived	Optional	Permitted Values
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
completion_locator_id	number(15)		x		
coproducts_supply	number		x	x	
net_quantity	number		x	x	Should be less than start quantity
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten
split_has_update_assy	number				Ignored

Split Transaction:

SPLIT WSM_SPLIT_MERGE TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			
transaction_type_id	number(15)	= 1			
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	=1 (PENDING)			
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
xml_document_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten

SPLIT WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
wip_entity_id	number				either wip_entity_id or wip_entity_name must be entered
xml_document_id	varchar2(240)				
wip_entity_name	varchar2(240)				either wip_entity_id or wip_entity_name must be entered
organization_id	number				
operation_seq_num	number	x			
intraoperation_step	number	x			1 or 3
representative_flag	varchar2(1)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	= 1 (PENDING)			

Column	Type	Required	Derived	Optional	Permitted Values
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
routing_seq_id	number				
primary_item_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten

SPLIT WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
description	varchar2(240)				
primary_item_id	number	x			To split and update assembly, specify new primary_item_id, else using starting lot's
class_code	varchar2(10)		x		Should be the same as starting job
start_quantity	number	x			
bom_reference_id	number				To be specified for non-standard jobs. Can specify new reference for split and update assembly
routing_reference_id	number				To be specified for non-standard jobs. Can specify new reference for split and update assembly

Column	Type	Required	Derived	Optional	Permitted Values
common_bom_sequence_id	number		x		From primary_item_id (for standard jobs) or bom_reference_id (for non-standard jobs) and alternate_bom_designator
common_routing_sequence_id	number		x		From primary_item_id (for standard jobs) or routing_reference_id (for non-standard jobs) and alternate_routing_designator
routing_revision	varchar2(3)		x	x	Can apply new value for split and update assembly
routing_revision_date	date		x	x	Can apply new value for split and update assembly
bom_revision	varchar2(3)		x	x	Can apply new value for split and update assembly

Column	Type	Required	Derived	Optional	Permitted Values
bom_revision_date	date			x	Can apply new value for split and update assembly
alternate_bom_designator	varchar2(10)		x	x	Can apply new value for split and update assembly but must also change primary_item_id
alternate_routing_designator	varchar2(10)		x	x	Can apply new value for split and update assembly but must also change primary_item_id
completion_subinventory	varchar2(10)		x		
starting_operation_code	varchar2(4)				

Column	Type	Required	Derived	Optional	Permitted Values
starting_operation_seq_num	number	x (For split and update, if job is (a) at Queue_intra operation or (b) at To Move intra operation and value of site level profile "WSM: Create Job Level BOM and Routing Copies" is YES)	x (For split and update, if job is at Queue intra operation and there exists only one matching operation in the target routing)	x (For split and update, if job is (a) at Queue intra operation and there exists only one matching operation in the target routing or (b) at To Move intra operation and value of site level profile "WSM: Create Job Level BOM and Routing Copies" is NO	For split, derived from the starting lots. For split and update assembly, specify starting operation on target routing
starting_std_op_id	number				
starting_intra operation_step	number		x = 1 (QUEUE) / 3 (TO MOVE)	x	
scheduled_start_date	date	x			
scheduled_completion_date	date	x			
starting_wip_entity_rename	varchar2(240)				
forward_op_option	number		x = 4		Ignored

Column	Type	Required	Derived	Optional	Permitted Values
bonus_acct_id	number(15)				
demand_class	varchar2(30)				
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated

Column	Type	Required	Derived	Optional	Permitted Values
completion_locator_id	number(15)		x		
coproducts_supply	number		x		
net_quantity	number		x		Should be less than start quantity
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten
split_has_update_assy	number		x	x	Defaulted to zero (no update of assembly) if unspecified. Specify 1 to update assembly

Note: For a Split transaction, a minimum of two records will be inserted if all the quantities in the starting lots are split.

Merge Transaction:

MERGE WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			

Column	Type	Required	Derived	Optional	Permitted Values
transaction_type_id	number(15)	= 2			
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				Ignored
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	=1 (PENDING)			
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
xml_document_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten

MERGE WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			

Column	Type	Required	Derived	Optional	Permitted Values
wip_entity_id	number				either wip_entity_id or wip_entity_name must be entered
xml_document_id	varchar2(240)				
wip_entity_name	varchar2(240)				either wip_entity_id or wip_entity_name must be entered
organization_id	number				
operation_seq_num	number	x			Merge is NOT permitted at a Non-standard operation.
intraoperation_step	number	x			1 or 3
representative_flag	varchar2(1)	x			1 or 2. Only one record can have a value of 1 for a Merge Transaction
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	= 1 (PENDING)			

Column	Type	Required	Derived	Optional	Permitted Values
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)		x		System generated
program_application_id	number				
program_id	number(15)		x		
program_update_date	date				
routing_seq_id	number				
primary_item_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
internal_group_id	number				Not for use. For internal purposes only. Any Previous value will be overwritten

MERGE WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
description	varchar2(240)				
primary_item_id	number	x			
class_code	varchar2(10)		x		Should be the same as parent representative job
start_quantity	number	x			
bom_reference_id	number				Derived based on Representative Lot
routing_reference_id	number				Derived based on Representative Lot
common_bom_sequence_id	number		x		Derived based on Representative Lot
common_routing_sequence_id	number		x		Derived based on Representative Lot
routing_revision	varchar2(3)		x	x	Derived based on Representative Lot
routing_revision_date	date		x	x	Derived based on Representative Lot

Column	Type	Required	Derived	Optional	Permitted Values
bom_revision	varchar2(3)		x	x	Derived based on Representative Lot
bom_revision_date	date			x	Derived based on Representative Lot
alternate_bom_designator	varchar2(10)		x	x	Derived based on Representative Lot
alternate_routing_designator	varchar2(10)		x	x	Derived based on Representative Lot
completion_subinventory	varchar2(10)		x	x	Derived based on Representative Lot
starting_operation_code	varchar2(4)				Derived based on Representative Lot
starting_operation_seq_num	number				Derived based on Representative Lot
starting_std_op_id	number				Derived based on Representative Lot

Column	Type	Required	Derived	Optional	Permitted Values
starting_intra operation_ste p	number		x = 1 (QUEUE) / 3 (TO MOVE)	x	Derived based on Representativ e Lot
scheduled_st art_date	date	x			
scheduled_co mpletion_dat e	date	x			
starting_wip_ entity_renam e	varchar2(240)				
forward_op_ option	number				Ignored
bonus_acct_i d	number(15)				
demand_clas s	varchar2(30)				
process_statu s	number	= 1 (PENDING)			
error_messag e	varchar2(240)				
last_update_ date	date	x			
last_updated _by	number	x			
creation_date	date	x			
created_by	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
completion_locator_id	number(15)				Derived based on Representative Lot
coproducts_supply	number				
net_quantity	number				Should be less than the resulting job start quantity.
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

Column	Type	Required	Derived	Optional	Permitted Values
split_has_update_assy	number				Ignored

Note: For a merge, a transaction minimum of two records are inserted into this table. One of the records is a representative lot.

Update Assembly Transaction:

UPDATE ASSEMBLY WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			
transaction_type_id	number(15)	= 3			
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	=1 (PENDING)			

Column	Type	Required	Derived	Optional	Permitted Values
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
xml_document_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE ASSEMBLY WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
wip_entity_id	number				either wip_entity_id or wip_entity_name must be entered
xml_document_id	varchar2(240)				
wip_entity_name	varchar2(240)				either wip_entity_id or wip_entity_name must be entered
organization_id	number				
operation_seq_num	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
intraoperation_step	number	=Q/TM			
representative_flag	varchar2(1)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)		x		System Generated
program_application_id	number				System Generated

Column	Type	Required	Derived	Optional	Permitted Values
program_id	number(15)		x		System Generated
program_update_date	date				System Generated
routing_seq_id	number				
primary_item_id	number				New Lot controlled Assembly Item. Should have a valid Network Routing in WSMTI.organization_id.
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE ASSEMBLY WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	

Column	Type	Required	Derived	Optional	Permitted Values
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				
description	varchar2(240)				
primary_item_id	number	x			
class_code	varchar2(10)		x		Should be the same as starting job
start_quantity	number	x			
bom_reference_id	number				To be specified for non-standard jobs
routing_reference_id	number				To be specified for non-standard jobs

Column	Type	Required	Derived	Optional	Permitted Values
common_bom_sequence_id	number		x		From primary_item_id (for standard jobs) or bom_reference_id (for non-standard jobs) and alternate_bom_designator
common_routing_sequence_id	number		x		From primary_item_id (for standard jobs) or routing_reference_id (for non-standard jobs) and alternate_routing_designator
routing_revision	varchar2(3)		x	x	
routing_revision_date	date		x	x	
bom_revision	varchar2(3)		x	x	
bom_revision_date	date			x	
alternate_bom_designator	varchar2(10)		x	x	
alternate_routing_designator	varchar2(10)		x	x	

Column	Type	Required	Derived	Optional	Permitted Values
completion_subinventory	varchar2(10)		x		
starting_operation_code	varchar2(4)				
starting_operation_seq_num	number	x (If job is (a) at Queue intra-operation or (b) at To Move intra-operation and value of site level profile "WSM: Create Job Level BOM and Routing Copies" is YES	x (If job is at Queue inter-operation and there exists only one matching operation in the target routing)	x (If job is (a) at Queue inter-operation and there exists only one matching operation in the target routing or (b) at To Move intra-operation and value of site level profile "WSM: Create Job Level BOM and Routing Copies" is NO	
starting_std_op_id	number				
starting_intraoperation_step	number		x = 1 (QUEUE) / 3 (TO MOVE)	x	3 is disallowed, if the starting operation sequence num is the last operation in the target routing.
scheduled_start_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
scheduled_completion_date	date	x			
starting_wip_entity_rename	varchar2(240)				
forward_op_option	number				
bonus_acct_id	number(15)				
demand_class	varchar2(30)				
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
completion_locator_id	number(15)				
coproducts_supply	number				
net_quantity	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten
split_has_update_assy	number				Ignored

Update Routing Transaction:

UPDATE ROUTING WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			

Column	Type	Required	Derived	Optional	Permitted Values
transaction_type_id	number(15)	= 5			
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	=1 (PENDING)			
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 -15	varchar2(150)				Ignored
request_id	number(15)				
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
xml_document_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE ROUTING WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			System Generated

Column	Type	Required	Derived	Optional	Permitted Values
wip_entity_id	number				either wip_entity_id or wip_entity_name must be entered
xml_document_id	varchar2(240)				
wip_entity_name	varchar2(240)				either wip_entity_id or wip_entity_name must be entered
organization_id	number				
operation_seq_num	number	x			
intraoperation_step	number	=Q/TM			1 or 3
representative_flag	varchar2(1)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)		x		System Generated
program_application_id	number				System Generated
program_id	number(15)		x		System Generated
program_update_date	date				System Generated
routing_seq_id	number				
primary_item_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE ROUTING WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				
description	varchar2(240)				
primary_item_id	number	x			
class_code	varchar2(10)		x		Should be the same as starting job
start_quantity	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
bom_reference_id	number				To be specified for non-standard jobs
routing_reference_id	number				To be specified for non-standard jobs
common_bom_sequence_id	number		x		From primary_item_id (for standard jobs) or bom_reference_id (for non-standard jobs) and alternate_bom_designator
common_routing_sequence_id	number		x		From primary_item_id (for standard jobs) or routing_reference_id (for non-standard jobs) and alternate_routing_designator
routing_revision	varchar2(3)		x	x	
routing_revision_date	date		x	x	
bom_revision	varchar2(3)		x	x	

Column	Type	Required	Derived	Optional	Permitted Values
bom_revision_date	date			x	
alternate_bom_designator	varchar2(10)		x	x	
alternate_routing_designator	varchar2(10)	x	x	x	NULL value permitted if different from current designator
completion_subinventory	varchar2(10)		x		
starting_operation_code	varchar2(4)				
starting_operation_seq_num	number				Ignored if the starting lot is at TO_MOVE Intra-operation.
starting_std_op_id	number				
starting_intraoperation_step	number		x = 1 (QUEUE) / 3 (TO MOVE)	x	3 is disallowed, if the starting operation sequence num is the last operation in the target routing.
scheduled_start_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
scheduled_completion_date	date	x			
starting_wip_entity_rename	varchar2(240)				
forward_op_option	number				
bonus_acct_id	number(15)				
demand_class	varchar2(30)				
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
completion_locator_id	number(15)				
coproducts_supply	number			x	1 or 2
net_quantity	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten
split_has_update_assy	number				Ignored

Update Quantity Transaction:

UPDATE QUANTITY WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			

Column	Type	Required	Derived	Optional	Permitted Values
transaction_type_id	number(15)	= 6			
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	=1 (PENDING)			
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
xml_document_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE QUANTITY WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			

Column	Type	Required	Derived	Optional	Permitted Values
wip_entity_id	number				either wip_entity_id or wip_entity_name must be entered
xml_document_id	varchar2(240)				
wip_entity_name	varchar2(240)				either wip_entity_id or wip_entity_name must be entered
organization_id	number				
operation_seq_num	number	x			
intraoperation_step	number	=Q/TM			
representative_flag	varchar2(1)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)		x		System Generated
program_application_id	number		x		System Generated
program_id	number(15)		x		System Generated
program_update_date	date		x		System Generated
routing_seq_id	number				
primary_item_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE QUANTITY WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				
description	varchar2(240)				
primary_item_id	number	x			
class_code	varchar2(10)		x		Derived from Starting Lots
start_quantity	number	x			
bom_reference_id	number				Derived from Starting Lots

Column	Type	Required	Derived	Optional	Permitted Values
routing_reference_id	number				
common_bom_sequence_id	number		x		Derived from Starting Lots
common_routing_sequence_id	number		x		Derived from Starting Lots
routing_revision	varchar2(3)		x	x	Derived from Starting Lots
routing_revision_date	date		x	x	Derived from Starting Lots
bom_revision	varchar2(3)		x	x	Derived from Starting Lots
bom_revision_date	date			x	Derived from Starting Lots
alternate_bom_designator	varchar2(10)		x	x	Derived from Starting Lots
alternate_routing_designator	varchar2(10)		x	x	Derived from Starting Lots
completion_subinventory	varchar2(10)		x	x	Derived from Starting Lots
starting_operation_code	varchar2(4)				Derived from Starting Lots
starting_operation_seq_num	number				Derived from Starting Lots

Column	Type	Required	Derived	Optional	Permitted Values
starting_std_op_id	number				Derived from Starting Lots
starting_intraoperation_step	number		x = 1 (QUEUE) / 3 (TO MOVE)	x	Derived from Starting Lots
scheduled_start_date	date	x			
scheduled_completion_date	date	x			
starting_wip_entity_rename	varchar2(240)				
forward_op_option	number				
bonus_acct_id	number(15)	x			Valid account Id from the organization's set of books. Important for proper valuation of the job.
demand_class	varchar2(30)				
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 -15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
completion_locator_id	number(15)				Derived from Starting Lots
coproducts_supply	number				Derived from Starting Lots
net_quantity	number				Should be less than the start quantity in resulting jobs.

Column	Type	Required	Derived	Optional	Permitted Values
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten
split_has_update_assy	number				Ignored

Update Lot Name Transaction:

UPDATE LOT NAME WSM_SPLIT_MERGE_TXN_INTERFACE (WSMTI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s			
transaction_type_id	number(15)	=7			7
transaction_date	date	x			
organization_id	number(15)	x			
reason_id	number				
suspense_acct_id	number(15)				
transaction_reference	varchar2(240)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	

Column	Type	Required	Derived	Optional	Permitted Values
process_status	number	=1 (PENDING)			
transaction_id	number(15)		=wsm_split_merge_transactions_s		
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)				System Generated
program_application_id	number(15)				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated

Column	Type	Required	Derived	Optional	Permitted Values
xml_document_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE LOT NAME WSM_STARTING_JOBS_INTERFACE (WSJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
wip_entity_id	number				either wip_entity_id or wip_entity_name must be entered
xml_document_id	varchar2(240)				
wip_entity_name	varchar2(240)				either wip_entity_id or wip_entity_name must be entered
organization_id	number				

Column	Type	Required	Derived	Optional	Permitted Values
operation_seq_num	number	x			
intraoperation_step	number	=Q/TM			
representative_flag	varchar2(1)				
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 - 15	varchar2(150)				Ignored
request_id	number(15)		x		System Generated

Column	Type	Required	Derived	Optional	Permitted Values
program_application_id	number				System Generated
program_id	number(15)		x		System Generated
program_update_date	date				System Generated
routing_seq_id	number				
primary_item_id	number				
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten

UPDATE LOT NAME WSM_RESULTING_JOBS_INTERFACE (WRJI)

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number(15)	=wsm_sm_txn_interface_s & =WSMTI.header_id			
group_id	number(15)		=wsm_sm_txn_int_group_s, if NULL	x	

Column	Type	Required	Derived	Optional	Permitted Values
wip_entity_name	varchar2(240)	x			Length cannot exceed 30 characters excluding sector extensions and lot separators
xml_document_id	varchar2(240)				
job_type	number				
organization_id	number				
description	varchar2(240)				
primary_item_id	number	x			
class_code	varchar2(10)		x		Derived from Starting Lots
start_quantity	number	x			
bom_reference_id	number				Derived from Starting Lots
routing_reference_id	number				Derived from Starting Lots
common_bom_sequence_id	number		x		Derived from Starting Lots
common_routing_sequence_id	number		x		Derived from Starting Lots

Column	Type	Required	Derived	Optional	Permitted Values
routing_revision	varchar2(3)		x	x	Derived from Starting Lots
routing_revision_date	date		x	x	Derived from Starting Lots
bom_revision	varchar2(3)		x	x	Derived from Starting Lots
bom_revision_date	date			x	Derived from Starting Lots
alternate_bom_designator	varchar2(10)		x	x	Derived from Starting Lots
alternate_routing_designator	varchar2(10)		x	x	Derived from Starting Lots
completion_subinventory	varchar2(10)		x	x	Derived from Starting Lots
starting_operation_code	varchar2(4)				
starting_operation_seq_num	number				
starting_std_op_id	number				
starting_intraoperation_step	number		x = 1 (QUEUE) / 3 (TO MOVE)	x	Derived from Starting Lots
scheduled_start_date	date	x			

Column	Type	Required	Derived	Optional	Permitted Values
scheduled_completion_date	date	x			
starting_wip_entity_rename	varchar2(240)				Not used
forward_op_option	number				Ignored
bonus_acct_id	number(15)				
demand_class	varchar2(30)				
process_status	number	= 1 (PENDING)			
error_message	varchar2(240)				
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
created_by	number	x			
last_update_login	number				
attribute_category	varchar2(30)				Ignored
attribute1 -15	varchar2(150)				Ignored

Column	Type	Required	Derived	Optional	Permitted Values
request_id	number(15)				System Generated
program_application_id	number				System Generated
program_id	number(15)				System Generated
program_update_date	date				System Generated
completion_locator_id	number(15)				Derived from Starting Lots
coproducts_supply	number				Derived from Starting Lots
net_quantity	number				Derived from Starting Lots
internal_group_id	number				Not for use. For internal purposes only. Any previous value will be overwritten
split_has_update_assy	number				Ignored

WSM_SERIAL_TXN_INTERFACE Table

Column	Type	Required	Derived	Optional	Permitted Values
header_id	number	Y			Header ID of the transaction record in (WSM_SPLIT_MERGE_TXN_INTERFACE)
transaction_type_id	number	Y			Transaction type id 3 – WIP Lot transactions
serial_number	varchar2			Y	Serial Number
generate_serial_number	number			Y	1 - To generate serial numbers qty provided by generate for qty. If this column is given, then values in serial_number and action_flag are ignored. (For Update Quantity Transaction only)

Column	Type	Required	Derived	Optional	Permitted Values
generate_for_qty	number			Y	Number of serials to be generated and associated. This column value is considered only when generate_serial_number column is populated with 1. (For Update Quantity Transaction only)
action_flag	number			Y	Action to be performed on the specified serial number mentioned in serial_number column. 1 - Add, 3 - Update (For Update Quantity Transaction only)
changed_wip_entity_name	varchar2			Y	The resulting job name for the serial number (For Split Transaction only)

Column	Type	Required	Derived	Optional	Permitted Values
serial_attribute_category cattribute1-20, dattribute1-10, nattribute1-10 (Unnamed columns – Serial attributes)				Y	(Serial Attributes Columns)
status_id time_since_new cycles_since_new time_since_overhaul cycles_since_overhaul time_since_repair cycles_since_repair time_since_visit cycles_since_visit time_since_mark cycles_since_mark number_of_repairs (named columns – serial attributes)	number			Y	(Serial Attributes Columns)
attribute_category, attribute1-15				Y	(Serial Descriptive flex field columns)

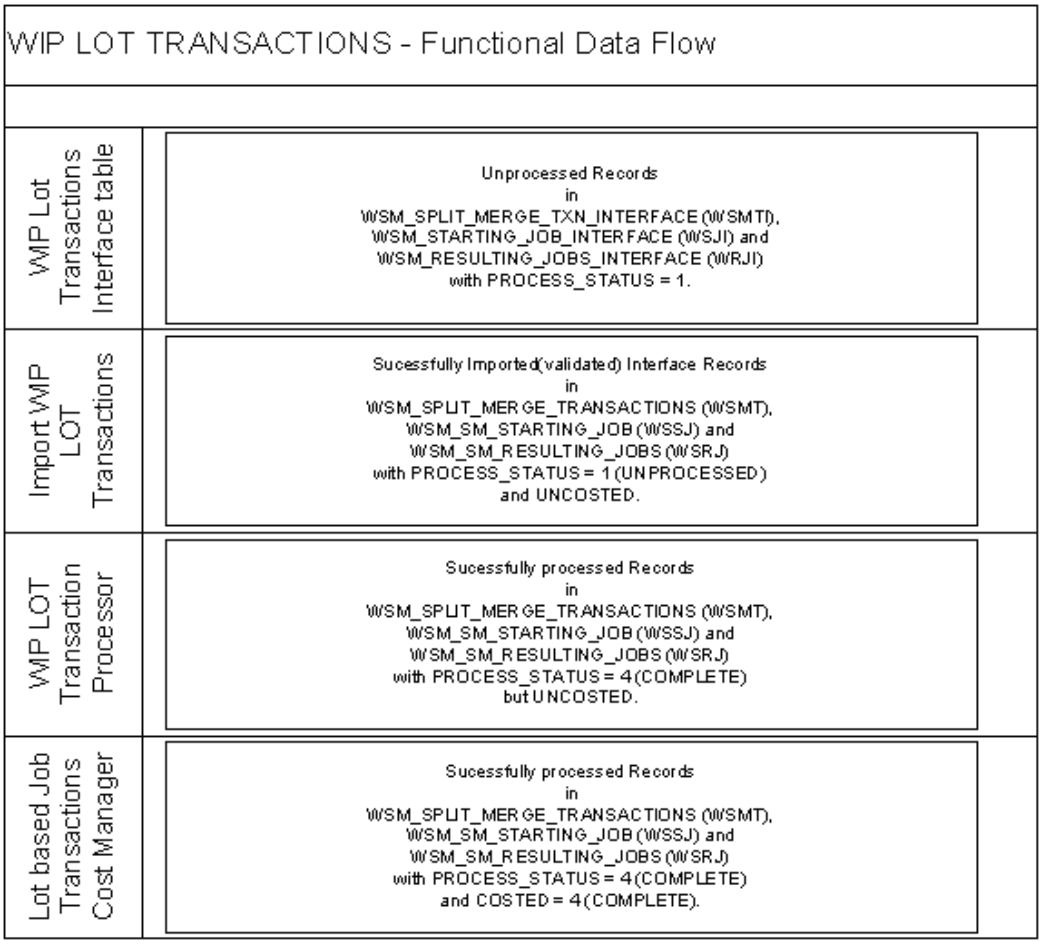
Upgrade to Patchset I

Prior to Patchset I, Importing WIP Lot Transactions was a three-step process

- Import WIP Lot Transactions
- WIP Lot Transactions Processor
- Lot Based Transaction Cost Manager

The following diagram illustrates the basic flow of import WIP Lot Transactions Concurrent Program before Patchset I.

Functional Data Flow of WIP Lot Transactions



From Patchset I onwards, it is a two-step process.

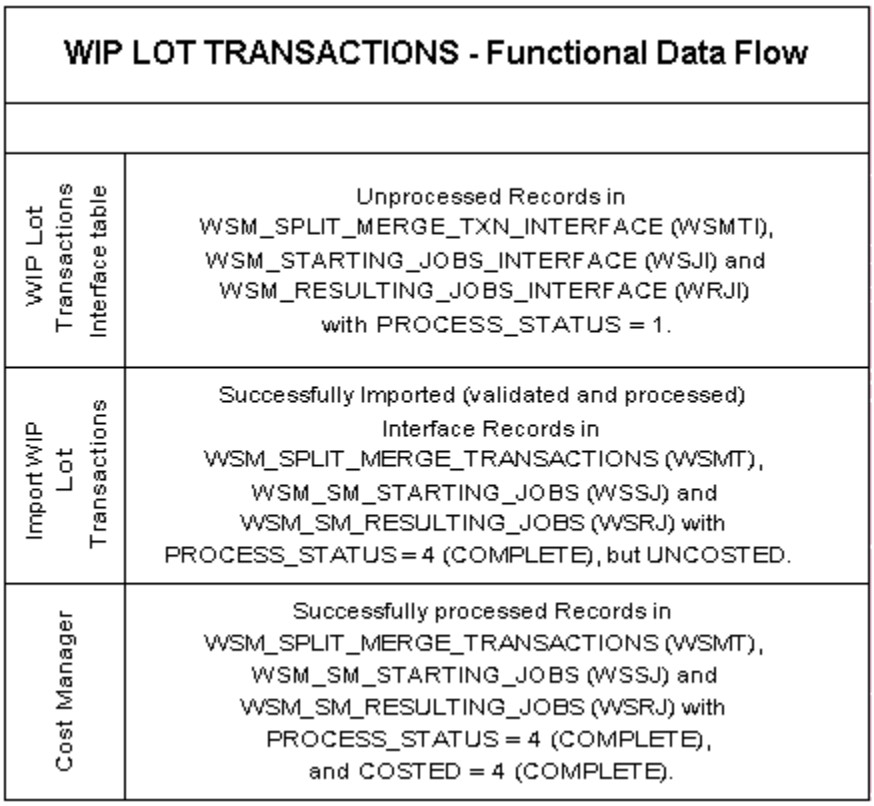
- Import WIP Lot Transactions
- Cost Manager

The Import WIP Lot Transactions Concurrent Program will pickup the transactions submitted in the Interfaces tables, one by one, by the order of transaction date. Each transaction is validated and processed immediately. If the validations as well as the processing completes sucessfully, the PROCESS_STATUS is set to COMPLETE, else the interface transaction errors with the appropriate message populated in ERROR_MESSAGE column. Thus, the additional step of running the WIP Lot Transactions Processor prior to Patchset I has been eliminated.

If there is any ERRORed transaction for a lot based job in the WIP Lot Transactions interface tables, the further PENDING transactions (with transaction dates after the ERRORed transaction) for this lot based job will not be picked up for processing. In such a case, the ERRORed transaction should be corrected and resubmitted with

PENDING status.

The following diagram illustrates the basic flow of import WIP Lot Transactions Concurrent Program from Patchset I.



Inventory Lot Transactions Interface Concurrent Program

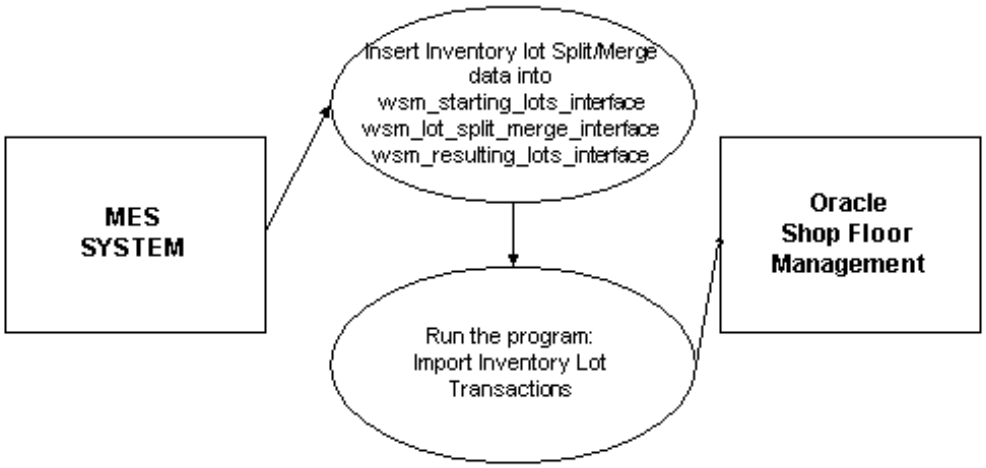
The Inventory Lot Transactions interface supports the following types of transactions:

Transaction Types and Descriptions

Transaction Type	Transaction Description
1	Split—creation of new Inventory lots
2	Merge—increases quantity of an existing Inventory lot by combining 2 or more lots
3	Translate—updates item, lot name, or both
4	Transfer—creates a subinventory transfer

Note: All of these transactions are identified as Miscellaneous Issues and Receipts in inventory material transactions.

The following diagram illustrates how data is imported from MES to Oracle Shop Floor Management through interface tables.



Calling the Import Inventory Lot Transactions Interface Program

In Oracle Shop Floor Management, the Run Requests menu, select Import Inventory Lot Transactions.

To run from the command line, enter the following command:

WSMPINVL.process_interface_rows (errbuf OUT VARCHAR2,

retcode OUT NUMBER,

p_group_id IN NUMBER,

p_header_id IN NUMBER,

p_mode IN NUMBER);

When calling this API, p_mode = 2 should be used to process all rows corresponding to group_id = p_group_id and row corresponding to header_id = p_header_id. Usually either p_group_id or p_header_id should be populated but both can be simultaneously specified.

WSM_LOT_SPLIT_MERGES_ INTERFACE Table

The following table lists the columns in the WSM_LOT_SPLIT_MERGES_ INTERFACE table and provides their load/update type and validation information. Some columns are not used by the program and marked as ignored.

WSM_LOT_SPLIT_MERGES_INTERFACE

Column	Null	Data Type	Required	Derived	Optional
transaction_id	yes	number		y (wsm_split_merge_transactions_s.nextval	
transaction_type_id	no	number	x		1 - split 2 - merge 3 - translate 4 - transfer
organization_id	no	number	x		
wip_flag	no	number	x		IGNORED
split_flag	no	number	x		IGNORED
last_update_date	no	date	x		
last_updated_by	no	number	x		
creation_date	no	date	x		
created_by	no	number	x		
transaction_reference	yes	varchar2(240)			
reason_id	yes	number			y
transaction_date	yes	date	x		
last_update_login	yes	number			y (FND_Global.Login_id)

Column	Null	Data Type	Required	Derived	Optional
attribute_category	yes	varchar2(30)			Ignored
attribute1 - 15	yes	varchar2(150)			Ignored
request_id	yes	number		y (FND_Global.conc_request_id)	System Generated
program_application_id	yes	number		y (FND_GLOBAL.prog_appl_id)	System Generated
Program_id	yes	number		y (FND_GLOBAL.conc_program_id)	System Generated
program_update_date	yes	date			y (SYSDATE)
process_status	yes	number	x		1 - pending 2 - running 3 - error 4 - complete
error_message	yes	varchar2(2000)			
group_id	yes	number	x		System Generated if NULL
transaction_reason	yes	varchar2(30)			
header_id	no	number	x		

WSM_STARTING_LOTS_INTERFACE Table

The following table lists the columns in the WSM_STARTING_LOTS_ INTERFACE table and provides their load/update type and validation information:

WSM_STARTING_LOTS_INTERFACE

Column	Null	Data Type	Required	Derived	Optional
transaction_id	yes	number		y (wsm_split_merge_transactions_s.nextval	
lot_number	no	varchar2(30)	x		
inventory_item_id	no	number	x		
organization_id	no	number	x		
quantity	no	number	x		
subinventory_code	no	varchar2(10)	x		
locator_id	yes	number	x (conditional)		if subinventory is locator controlled
revision	yes	varchar2(3)			y
last_update_date	no	date	x		
last_updated_by	no	number	x		
creation_date	no	date	x		
created_by	no	number	x		

Column	Null	Data Type	Required	Derived	Optional
last_update_login	yes	number			y (FND_Global.Login_id)
attribute_category	yes	varchar2(30)			Ignored
attribute1 - 15	yes	varchar2(150)			Ignored
request_id	yes	number		y (FND_GLOBAL.conc_request_id)	System Generated
program_application_id	yes	number		y (FND_GLOBAL.prog_appl_id)	System Generated
program_id	yes	number		y (FND_GLOBAL.conc_program_id)	System Generated
program_update_date	yes	date			y (SYSDATE)
header_id	no	number	x		

WSM_RESULTING_LOTS_INTERFACE Table

The following table lists the columns in the WSM_RESULTING_LOTS_ INTERFACE table and provides their load/update type and validation information:

WSM_RESULTING_LOTS_INTERFACE

Column	Null	Data Type	Required	Derived	Optional
transaction_id	yes	number		y (wsm_split_merge_transactions_s.nextval	
lot_number	no	varchar2(30)	x		
inventory_item_id	no	number	x		
organization_id	no	number	x		
wip_entity_id	yes	number		x	used only by mode 2 - lot controlled for imported lot based jobs
quantity	no	number	x		
subinventory_code	no	varchar2(10)	x		
locator_id	yes	number	x (conditional)		if subinventory is locator controlled
revision	yes	varchar2(3)			y
last_update_date	no	date	x		
last_updated_by	no	number	x		
creation_date	no	date	x		
created_by	no	number	x		

Column	Null	Data Type	Required	Derived	Optional
last_update_login	yes	number			y (FND_Global.Login_id)
attribute_category	yes	varchar2(30)			Ignored
attribute1 - 15	yes	varchar2(150)			Ignored
request_id	yes	number		y (FND_GLOBAL.conc_request_id)	System Generated
program_application_id	yes	number		y (FND_GLOBAL.prog_appl_id)	System Generated
program_id	yes	number		y (FND_GLOBAL.conc_program_id)	System Generated
program_update_date	yes	date			y (SYSDATE)
header_id	no	number	x		

Validation of Inventory Lot Transactions

The following describes validation of inventory lot transactions.

Standard Validation

Oracle Shop Floor Management validates all required columns in the Inventory Lot Transactions interface. For specific information on the data implied by these columns, see your Oracle Shop Floor Management Technical Reference Manual for details.

Error Handling

If any validation fails, the program will return error status to the calling module. The Inventory Lot Transactions processes the rows and reports the following values for every record.

Error Handling Table

Condition	PROCESS_STATIS	ERROR_MESSAGE
Success	4	null
Failure	3	actual error message
Running	2	null

See Also

Oracle Shop Floor Management Technical Reference Manual

Functional Overview

Following are the major design features of the interface:

1. Insert rows into the WSM_LOT_SPLIT_MERGE_INTERFACE, WSM_STARTING_LOTS_INTERFACE, WSM_RESULTING_LOTS_INTERFACE tables. Transactions in one of these tables are joined with the transactions in the other two tables through the HEADER_ID column.

2. Use the wsm_lot_sm_ifc_header sequence to populate the header_id and group_id columns; header_id column must be NOT NULL.

3. You can group the transactions to be put into the interface table by providing a GROUP_ID.

4. When a concurrent request is launched for importing Inventory Lot Transactions, you can specify a GROUP_ID. If a GROUP_ID is specified, then only those records in the group and in the Pending status would by considered for processing. If no group is specified while launching the import program, then the processing is done group by group for all the pending records in the respective group. Unique GROUP_IDs are assigned to records in which the GROUP_ID is null.

5. Within a group, transactions are processed in the order of TRANSACTION_DATE.

6. Inventory lot transaction is not supported for lots with serial controlled assemblies.

Errors and Validations

1. The interface makes all the necessary validations that would be typically done from the front-end.

2. Whenever one record errors out within a group, the PROCESS_STATUS of the

corresponding header id is set to ERROR.

3. Typically, interdependent transactions should not be grouped together. It is preferable to group only independent transactions.

4. If a GROUP_ID is not specified, then the import program would assign a GROUP_ID to each and every transaction based on the TRANSACTION_DATE.

5. If the import program is run without specifying a GROUP_ID, and if some GROUP_ID errors out, then the status of the concurrent request is set to WARNING. But if a group is specified and it errors out, then the status of the concurrent request is set to ERROR.

6. All errors and warnings are written to the WSM_INTERFACE_ERRORS table.

Lot Attributes

You can process lot attributes of a job while processing any of the following functions:

1. Create, update lot based job

2. WIP lot transactions

3. Move transactions

4. Lot creations

5. Inventory Lot Transactions

You can continue to process all of the above the same way they are described in this manual. In addition, you may insert lot attribute values into the mtl_transaction_lots_interface table to process the lot attributes. There are two sets of lot attributes:

1. WMS Attributes: Comprises C_attributes1 through C_attributes15, D_attributes1 through D_attributes10, N_attributes1 through N_attributes10, and the named attributes such as Place_of_origin and Length.

2. INV Attributes: Comprises of Attribute1 through Attribute15

Errors and Validations

Errors and warnings, if any, are written into the WS M_INTERFACE_ERRORS table.

Processing Lot Attributes

Perform the following steps to populate lot attributes into the interface table and process them:

1. Populate the transaction_interface_id using the sequence mtl_material_transactions_s.

2. Populate Product_transaction_id with the same value that is populated in header_id from the respective sequence mentioned in the following table for each job, lot creation and transaction.

Transaction/Lot Job Creation	Interface table	Column name	Sequence
Lot based job creation	WSM_LOT_JOB_INTERFACE	header_id	wsm_lot_sm_ifc_header_s
WIP Lot Transactions	WSM_RESULTING_JOBS_INTERFACE	header_id	wsm_sm_txn_interface_s
Move transactions	WSM_RESULTING_JOBS_INTERFACE	header_id	wsm_lot_move_txn_interface
Lot creation	WSM_LOT_JOB_INTERFACE	header_id	wsm_lot_sm_ifc_header_s

3. Based on the descriptive flex field definitions that you or your system administrator may have defined for Lot Attributes, fill in the appropriate column values of Lot_attributes_category, C_Attribute1 through C_Attribute15, D_attribute1 through D_attribute10, N_attribute1 through N_attribute10 and the named attributes such as Place_of_origin and Length. These columns need to be filled in only if WMS is installed.

4. Based on the descriptive flex field definitions that you or your system administrator has defined for Maintain Lot Number, fill in the appropriate values of Attribute_category and Attribute1 through Attribute15.

The following rules apply while processing the lot attributes:

1. If lot attributes for a given lot already exist, then the attributes will be updated with the attribute values from the interface table for the lot.

2. If lot attributes for a given lot does not exist, then the attributes will be created with the attribute values from the interface table for the lot.

3. While processing Lot Creation or WIP lot transactions, the attributes for the resulting lots are inherited based on the following rules in the specified order:
   - If you have inserted lot attributes in the interface for the lot, then attributes are

inserted or updated with the values from the interface for the lot.

- If attributes for the resulting lot already exist, then the attributes are not inherited from the source lot.
- If context of the Source and resulting lots are the same, then WMS attributes are inherited. If not, WMS attributes are not inherited.
- Inventory attributes are inherited from the source lot whether or not the context of the source and resulting lots match.

If you intend to update the lot attributes of a resulting lot, a record must be inserted into the MTL_TRANSACTION_LOTS_INTERFACE as follows:

1. product_transaction_id - Insert the header_id of the corresponding inventory lot transaction record in WSM_LOT_SPLIT_MERGES_INTERFACE table.
2. product_code - 'WSM'
3. lot_number - Insert the corresponding resulting lot in WSM_RESULTING_LOTS_INTERFACE table
   - If the transaction type is Translate, the resulting lot item context is different from the starting lot item, and there are mandatory attributes defined for the resulting lot items, then you must provide values for these mandatory lot attributes.
   - If the transaction Type is Merge and there is no corresponding record in the MTL_TRANSACTION_LOTS_INTERFACE table for a resulting lot/resulting job, then lot attributes will be copied from the representative lot/representative job while processing for inventory lot transaction/WIP lot transaction.

MTL_TRANSACTION_LOTS_INTERFACE

Column	Type	Required	Derived	Optional	Permitted Values
Transaction_interface_id	number	x			mtl_material_transactions_s.NEXTVAL
product_transaction_id	number	x	=wsm_sm_txn_int_group_s, if NULL	x	As mention in table 9-40
product_code	varchar2(5)	x			WSM

Column	Type	Required	Derived	Optional	Permitted Values
lot_number	varchar2(30)	x			
transaction_quantity	number	x			Ignored. This value is required because it is a NOT NULL column
last_update_date	date	x			
last_updated_by	number	x			
creation_date	date	x			
creation_by	number	x			
last_update_login	number	x		x	
request_id	number			x	
program_application_id	number			x	
program_id	number			x	
program_update_date	date			x	

11

Oracle Quality Open Interfaces and APIs

This chapter covers the following topics:

- Collection Import Interface
- Collection Plan ViewsCollection plan views
- Overview of the Oracle Quality Application Program Interfaces
- Create Specification Application Program Interface
- Add Specification Element Application Program Interface
- Complete Specification Application Program Interface
- Delete Specification Application Program Interface
- Delete Specification Element Application Program Interface
- Copy Specification Application Program Interface
- Using the Specification APIs
- Create Collection Plan Application Program Interface
- Add Collection Plan Element Application Program Interface
- Complete Collection Plan Processing Application Program Interface
- Delete Collection Plan Application Program Interface
- Delete Collection Plan Element Application Program Interface
- Copy Collection Plan Application Program Interface
- Using the Collection Plan APIs

Collection Import Interface

You can use the Collection Import Interface to add new quality results data or to update existing quality results data in the Quality data repository. For example, you can load data from sources such as test equipment and gauges into the Collection Import

Interface Table, then import them into the Quality data repository. Since Collection Import works as a background process, the flow of your work is not interrupted.

Functional Overview

The Collection Import process consists of the following three major steps:

Loading the Collection Import Interface Table

Before you can import quality results data, you must load it into the Collection Import Interface Table. The programming languages and tools that you use to load this data are highly dependent on your data source.

Launching the Collection Import Manager

After you load data into the Collection Import Interface table, you launch the Collection Import Manager. The Collection Import Manager is a background process that searches the interface for new rows. If any are found, it launches one or more Import Workers, which validate the data. It then inserts valid records, or updates existing records in the Quality results data repository (QA_RESULTS), and invokes any actions associated with these records.

The Collection Import Interface Table can contain multiple rows. Each row specifies either that a new record is to be added or an existing record is to be updated in the Quality data repository. When the Collection Import Manager completes a transaction, it either updates records in the Quality data repository, or inserts all records specified as Insert into it. The Collection Import Manager can only perform one *type* of transaction (updating or inserting records) each time that it completes a transaction, although it can perform that transaction on multiple records (rows).

You specify the type of transaction to be performed in the Transaction Type field, which appears when you launch the Collection Import Manager. This field can take one of two values: Insert Transaction or Update Transaction.

> **Note:** The Collection Import Manager executes all types of actions, except the Display a message to the operator action, which must be processed online. Records that are associated with the Reject the input action are not imported into the Quality results data repository.

Rows that fail validation are marked and remain in the Collection Import Interface Table. Error messages explaining why records failed validation or processing are inserted into the Errors table.

Updating Collection Import

The final step in the Collection Import process is viewing, updating, and resubmitting failed rows. You use the Update Collection Import form to optionally delete any records that you do not want to resubmit.

Note: Do not confuse this step with running the Collection Import Manager in the "Update Transaction" mode.

See Also

Collection Import Interface Table

Example: Collection Import SQL Script

Collection Import Manager

Updating Collection Import, *Oracle Quality User's Guide*

Collection Import Interface Table

The Collection Import Interface Table (QA_RESULTS_INTERFACE) is similar in structure to the Quality results database table (QA_RESULTS), however, it contains a number of additional columns.

The following table describes the columns in the Collection Import Interface table.

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
CHARACTER1 through CHARACTER100	Varchar(150)					x
COLLECTION_ID	Number(38)			x		
COMP_GEN_LOC_CTRL_CODE	Number		x1			
COMP_ITEM	Varchar2(2000)					x
COMP_ITEM_ID	Number		x			x

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
COMP_LOCATION_CONTROL_CODE	Number		x1			
COMP_LOCATOR	Varchar2(2000)					x
COMP_LOCATOR_ID	Number		x			x
COMP_LOT_NUMBER	Varchar2(30)					x
COMP_RESTRICT_LOCATORS_CODE	Number		x1			
COMP_RESTRICT_SUBINV_CODE	Number		x1			
COMP_REVISION	Varchar2(3)					x
COMP_REVISION_QTY_CONTROL_CODE	Number		x1			
COMP_SERIAL_NUMBER	Varchar2(30)					x
COMP_SUB_LOCATOR_TYPE	Number		x1			

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
COMP_SUBINVENTORY	Varchar2(10)					x
COMP_UOM	Varchar2(3)					x
CREATED_BY	Number		x1			
CREATION_DATE	Date		x1			
CUSTOMER_ID	Number		x			x
CUSTOMER_NAME	Varchar2(50)					x
DEPARTMENT	Varchar2(10)					x
DEPARTMENT_ID	Number		x			x
FROM_OP_SEQ_NUM	Number					x
GEN_LOC_CTRL_CODE	Number		x1			x
GROUP_ID	Number		x1			
INSERT_TYPE	Number				x	
ITEM	Varchar2(2000)					x

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
ITEM_ID	Number		x			x
JOB_NAME	Varchar2(240)					x
LAST_UPDATE_DATE	Date		x			
LAST_UPDATE_LOGIN	Number		x1			
LAST_UPDATED_BY	Number		x1			
LINE_ID	Number		x			x
LOCATION_CONTROL_CODE	Number		x1			x
LOCATOR	Varchar2(2000)					x
LOCATOR_ID	Number		x			x
LOT_NUMBER	Varchar2(30)					x
MARKER	Number		x1			
MATCHING ELEMENTS	Varchar2(1000)				x	
ORGANIZATION_CODE	Varchar2(3)	x				

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
ORGANIZATION_ID	Number		x			
PLAN_ID	Number		x			
PLAN_NAME	Varchar2(30)	x				
PO_AGENT_ID	Number		x			
PO_HEADER_ID	Number		x			x
PO_LINE_NUM	Number					x
PO_NUMBER	Varchar2(20)					x
PO_RELEASE_ID	Number			x		x
PO_RELEASE_NUM	Number		x			x
PO_SHIPMENT_NUM	Number					x
PO_TYPE_LOOKUP	Varchar2(25)		x			x
PROCESS_STATUS	Number	x				
PRODUCTION_LINE	Varchar2(10)					x

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
PROGRAM_APPLICATION_ID	Number		x1			
PROGRAM_ID	Number		x1			
PROGRAM_UPDATE_DATE	Date		x1			
PROJECT_ID	Number		x			x
PROJECT_NUMBER	Varchar2(25)		x			x
QA_CREATED_BY	Number		x			
QA_CREATED_BY_NAME	Varchar2(100)			x		
QA_LAST_UPDATED_BY	Number		x			
QA_LAST_UPDATED_BY_NAME	Varchar2(100)			x		
QUANTITY	Number					x
RECEIPT_NUM	Varchar2(30)					x
REQUEST_ID	Number		x1			

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
RESOURCE_CODE	Varchar2(10)					x
RESOURCE_ID	Number		x			x
RESTRICT_LOCATORS_CODE	Number		x1			x
RESTRICT_SUBINV_CODE	Number		x1			x
REVISION	Varchar2(3)					x
REVISION_QTY_CONTROL_CODE	Number		x1			x
RMA_HEADER_ID	Number		x			x
RMA_NUMBER	Number					x
SALES_ORDER	Number					x
SERIAL_NUMBER	Varchar2(30)					x
SO_HEADER_ID	Number		x			x
SOURCE_CODE	Varchar2(30)				x	

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
SOURCE_LINE_ID	Number				x	
SPEC_ID	Number		x			
SPEC_NAME	Varchar2(30)				x	
STATUS	Varchar2(25)		x			x
SUB_LOCATOR_TYPE	Number		x1			x
SUBINVENTORY	Varchar2(10)					x
TASK_ID	Number		x			x
TASK_NUMBER	Varchar2(25)		x			x
TO_DEPARTMENT	Number					x
TO_DEPARTMENT_ID	Varchar2(10)		x			x
TO_OP_SEQ_NUM	Number					x
TRANSACTION_DATE	Date		x			
TRANSACTION_INTERFACE_ID	Number			x		

Column Name	Data Type	Required	Derived (Leave Null)	Derived or UserOptional	Optional	Plan Specific
UOM	Varchar2(3)					x
VALIDATE_FLAG	Number				x	
VENDOR_ID	Number		x			x
VENDOR_NAME	Varchar2(80)					x
WIP_ENTITY_ID	Number		x			x

[1] These columns must be left null in all circumstances.

Derived Data

The Collection Import Manager derives data for some columns in the Collection Import Interface Table using foreign key relationships within Oracle Manufacturing. You can, however, insert user-defined data into some derived columns.

Control Columns

Control columns store information specific to the import process. These columns include:

TRANSACTION_INTERFACE_ID: Each row added to the Collection Import Interface Table receives a unique Transaction Interface ID.

> **Note:** You should leave this field empty.

PROCESS_STATUS: The Process Status identifies the state of the transaction and has four possible values:

- 1 Pending
- 2 Running
- 3 Error

- 4 Completed

When loading records into the Collection Import Interface Table, you must assign them an initial process status of 1 (Pending). During validation, the Collection Import Manager updates the status to 2 (Running). Rows that fail validation are assigned a status of 3 (Error). Successfully validated rows are assigned a status of 4 (Completed) and are immediately deleted from the interface table.

> **Note:** You can prevent status 4 (Completed) rows from being deleted from the Collection Import Interface table by setting the Oracle Master Scheduling/MRP and Supply Chain Planning *MRP:Debug Mode* profile option to Yes (see the *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning User's Guide* for more information).

VALIDATE_FLAG: The Validate Flag determines whether the Collection Import Manager validates records in the Collection Import Interface table before importing them into the Quality results database table. The Validate Flag is present in the QA_RESULTS_INTERFACE table; however, it is not present in the import view. There are two values for this field:

- 1 Yes
- 2 No

Normally this flag is assigned a value of 1. When set to 1, or left blank, records are validated. When set to 2, records are not validated.

> **Note:** It is potentially dangerous to turn the Validate Flag off during Collection Import. Without validation, inconsistent data is not rejected.

INSERT_TYPE: This field determines whether the Collection Import Manager will insert new records or update existing records in the Quality data repository. There are two values for this field:

- 1 Insert
- 2 Update

The default for the field is 1 Insert. When set to 1, or left blank, Collection Import inserts new records into the Quality data repository. When set to 2, it updates existing records in the repository.

MATCHING_ELEMENTS: This is a comma-separated list of column names. Collection Import uses these column names as search keys when it updates existing records in the Quality data repository. If an existing record has the same data in the columns listed by Matching Elements, the record is updated so that it is identical to the corresponding row in the Collection Import Interface table. If any of the record's columns in the table is set to NULL, they will not be updated in the Quality data repository. Also, an import

row in the table can match one and only one record in the repository; if the row in question has no match or has more than one match, Collection Import will reject it.

SPEC_NAME: This field determines what specification will be used to validate the record. The value in this field should be set to the name of a specification. If no specification is required, you should set this field to NULL. If the field is *null*, the specification is enforced at the collection element level. If the field is *not null*, the named specification is used.

PLAN_NAME: This field should contain the name of the collection plan. The name must be written in capital letters.

Who Columns

Collection Import uses the name of the current user to derive values for standard Who columns:

- QA_LAST_UPDATE_BY_NAME
- QA_CREATED_BY_NAME

You have the option to insert data into these derived Who columns. If you do so, the Collection Import Manager validates but does not override your data.

Self Service Columns

The Collection Import Interface imports data entered by suppliers through Oracle Supplier Management Portal's Outside Processing Workbench and Quality Plans for Shipments web pages. If any records fail validation during the import process, a workflow notifies the Buyer. The following column is used to derive the name of the Buyer who will receive the workflow notification:

- PO_AGENT_ID

Optional Data

All columns that contain user-defined, reference information, and predefined collection elements are optional.

> **Note:** When you load data into the interface table, you must enter the default values for the collection plan manually.

Name Columns

For every column in the Quality results database table (QA_RESULTS) that stores a foreign-key ID, like CUSTOMER_ID, the Collection Import Interface table contains two columns; one for the ID and one for the name. For example, customer data is associated with the CUSTOMER_ID and CUSTOMER_NAME columns in the interface table. You

should always enter data into the name fields. The ID fields are used by the Collection Import Worker during processing, and any values entered in these fields are ignored. There is, however, one exception. If you have set the VALIDATE_FLAG field to No (see below), you must enter the underlying IDs, since they are transferred directly into the results table without undergoing validation.

Source Columns

SOURCE_CODE, SOURCE_LINE_ID. These optional columns identify the sources of your Quality data. For example, if you are importing data that has been downloaded into an ASCII file as well as data from a data collection device, you can use a different source code to indicate the origin of each data record. To record more detailed information about the source, you can also fill in the source line ID. Keeping track of sources is often useful in tracking down validation problems.

Collection Import Results Database Views

Collection Import Results Database Views are created and updated when you create and update collection plans. Collection Import Results Database Views facilitate the insertion of data into the Collection Import Interface table. Instead of inserting data directly into the import table, insert data into views of the table.

The Collection Import Results Database View remaps the generic CHARACTERx columns to columns with meaningful names. If define the collection elements Defect Code and Inspector ID for a collection plan, the names of these collection elements are mapped to the CHARACTERx columns. The import view eliminates import table columns that represent collection elements not added to a collection plan. If you create a collection plan, but do not add the PO Number and PO Line Number collection elements, the corresponding PO_NUMBER and PO_LINE_NUM columns are not included in the import view.

See Also

Collection Import Manager

Creating Collection Plans, *Oracle Quality User's Guid*e

Collection Plan and Import Results Database Views, *Oracle Quality User's Guide*

Example: Collection Import SQL Script

Oracle Quality uses the naming convention for collection import results database views: Q_<collection-plan-name>_IV. Consider the collection plan IMPORT's collection elements:

- Defects
- Department

- From Ops Seq Number
- Item
- Job Number
- Lot Number
- Off Location
- Operator
- Revision
- Thickness
- To Ops Seq Number

When you create this collection plan, Oracle Quality automatically creates a collection import results database view called Q_IMPORT_IV. This is a view to the Collection Import Interface table QA_RESULTS_INTERFACE. This view contains the following columns:

SQL> DESCRIBE Q_IMPORT_IV;	Data Type
COLLECTION_ID	Number
DEFECTS	Varchar(150)
DEPARTMENT	Varchar2(10)
FROM_OP_SEQ_NUM	Number
INSERT_TYPE	Number
ITEM	Varchar2 (2000)
JOB_NAME	Varchar2(240)
LOT_NUMBER	Varchar2(30)
MATCHING_ELEMENTS	Varchar2(1000)
OFF_LOCATION	Varchar(150)

SQL> DESCRIBE Q_IMPORT_IV;	Data Type
OPERATOR	Varchar(150)
ORGANIZATION_CODE	Varchar2(3)
PLAN_NAME	Varchar2(30)
PROCESS_STATUS	Number
QA_CREATED_BY_NAME	Varchar2(100)
QA_LAST_UPDATED_BY_NAME	Varchar2(100)
REVISION	Varchar2(3)
SOURCE_CODE	Varchar2(30)
SOURCE_LINE_ID	Number
SPEC_NAME	Varchar2(30)
THICKNESS	Varchar(150)
TO_OP_SEQ_NUM	Number
TRANSACTION_INTERFACE_ID	Number

The following PL/SQL code demonstrates how you can insert collection import results directly into this view:

SQL> INSERT INTO Q_IMPORT_IV (

- PROCESS_STATUS,

  ORGANIZATION_CODE,

  PLAN_NAME,

  ITEM,

  REVISION,

  LOT_NUMBER,

  JOB_NAME,

  FROM_OP_SEQ_NUM,

TO_OP_SEQ_NUM,

DEPARTMENT,

OPERATOR,

DEFECTS,

THICKNESS,

OFF_LOCATION

)

VALUES (

1,

'MAS',

'IMPORT',

'ITEM8',

'0',

'A',

'DJ1',

10,

20,

'D1',

'jus',

'br',

'40',

'0'

);

The following PL/SQL code demonstrates how you can insert rows into the QA_RESULTS_INTERFACE table to update information in the Quality data repository. (Note that, in this example, the search keys 'ITEM = ITEM8', 'REVISION = '0', and 'LOT_NUMBER = 'A' are used to search for a matching record in the Quality data repository; then, this example modifies that matching record's DEFECTS column to equal 'bent' and THICKNESS to equal '45'. Because other columns are left to NULL, this update transaction leaves the record's corresponding fields unchanged in the repository.):

SQL>INSERT INTO Q_IMPORT_IV (

- PROCESS_STATUS,

  INSERT_TYPE,

```
MATCHING_ELEMENTS,
ORGANIZATION_CODE,
PLAN_NAME,
ITEM,
REVISION,
LOT_NUMBER,
DEFECTS,
THICKNESS,
)
VALUES (
1,
2,
'ITEM, REVISION, LOT_NUMBER',
'MAS'
'IMPORT',
'ITEM8',
'0'
'A',
'bent',
'45'
);
```

Collection Import Manager

The Collection Import Manager is a background concurrent process that checks the Collection Import Interface table for new records (rows). If there are new rows, it launches one or more Collection Import Worker processes. Worker processes carry out the three main phases of the import process:

- Validation
- Transfer
- Error handling

You can specify the maximum number of rows that you would like each worker process to handle when you launch the Collection Import Manager.

The Collection Import Manager can handle an unlimited number of rows, even though you set a maximum for each worker process. If additional rows are needed, the Collection Import Manager automatically launches new workers to handle more rows. For example, if you specify that you would like each worker process to handle a maximum of ten rows, but you submit 53 new records, the Collection Import Manager automatically launches six concurrent workers, the first five handle ten rows each, and the sixth handles the last three rows.

Validation

During the validation phase, each row in the Collection Import Interface is examined to verify that the data is valid and that required data is not missing. For example, rows are evaluated for context element dependencies, and rows that contain, for instance, serial numbers but not items, fail validation. See: Dependencies Between Context Elements and Actions, *Oracle Quality User's Guide*.

Successfully validated records are transferred to the Quality results database table (QA_RESULTS).

Transfer

During the transfer phase, Collection Import Workers insert successfully validated rows into the Quality results database table and delete them from the interface table.

Error Handling

Rows that fail validation remain in the Collection Import Interface table, and records detailing the errors are inserted into the Errors table (QA_INTERFACE_ERRORS).

Records can fail validation for several reasons:

- A mandatory collection element is left null
- A value cannot be converted to the correct data type (e.g. a value of 'abc' for pH, when pH is a number data type)
- A value is not in the set of lookup values defined for a collection element (e.g. a value of 40 for defect code, when defect code only has the values 10, 20, and 30 as lookups
- A value is not in the set of values contained in a foreign table (e.g. the value given for supplier ID is not found in the suppliers table)
- A value causes a Reject the Input action to be fired
- A value falls outside the reasonable limit range for the given specification (see: SPEC_NAME,).
- A value in a dependent field is not in the subset of values defined for the master

value (e.g. revision is 'C' when the master item only has 'A' and 'B' as possible revisions)

- A value is given for a collection element that is disabled on the collection plan

See Also

Collection Import Interface Table

Importing Quality Results Data, *Oracle Quality User's Guide*

Updating Collection Import, *Oracle Quality User's Guide*

Collection Plan Views

Collection Plan Views are created and updated when you create and update collection plans. Collection plan views allow you to query and inspect data for a particular collection plan in the Quality data repository.

The Collection Plan View remaps the generic CHARACTERx columns to columns with meaningful names. For example, if you have defined the collection elements Defect Code and Inspector ID for a collection plan, the names of these collection elements are automatically mapped to the CHARACTERx columns. The plan view also eliminates table columns that represent collection elements that have not been added to a collection plan. For example, if you create a collection plan, but do not add to it the PO Number and PO Line Number collection elements, the corresponding PO_NUMBER and PO_LINE_NUM columns are not included in the plan view.

Example

Oracle Quality uses the following naming convention for collection plan views: Q_<collection-plan-name>_V. For example, consider the following collection plan called WIP DEMO with the following collection elements:

- Defects
- Department
- From Ops Seq Number
- Item
- Job Number
- Lot Number
- Off Location
- Operator

- Revision
- Thickness
- To Ops Seq Number

When you create this collection plan, Oracle Quality automatically creates a collection plan view called Q_WIP_DEMO_V. This is a view to the Collection Results table QA_RESULTS. This view contains the following columns:

SQL> DESCRIBE Q_WIP_DEMO_V;	Data Type
COLLECTION_ID	Number
CREATED_BY	Varchar2 (100)
CREATED_BY_ID	Number
CREATION_DATE	Date
DEFECTS	Varchar (150)
DEPARTMENT	Varchar2 (10)
FROM_OP_SEQ_NUM	Number
ITEM	Varchar2 (2000)
JOB_NAME	Varchar2 (240)
LAST_UPDATE_DATE	Date
LAST_UPDATE_LOGIN	Number
LAST_UPDATED_BY	Varchar2 (100)
LAST_UPDATED_BY_ID	Number
LOT_NUMBER	Varchar2 (30)
OCCURRENCE	Number
OFF_LOCATION	Varchar (150)

SQL> DESCRIBE Q_WIP_DEMO_V;	Data Type
OPERATOR	Varchar (150)
ORGANIZATION_ID	Number
ORGANIZATION_NAME	Varchar2 (60)
PLAN_ID	Number
PLAN_NAME	Varchar2 (30)
REVISION	Varchar2 (3)
ROW_ID	Undefined
THICKNESS	Varchar (150)
TO_OP_SEQ_NUM	Number

Overview of the Oracle Quality Application Program Interfaces

The Oracle Quality APIs enable you to import your specification and collection plan information from a legacy or external system into Oracle Quality. The Oracle Quality APIs are divided into two groups:

- Specification APIs (PL/SQL package name: QA_SPECS_PUB)
- Collection Plan APIs (PL/SQL package name: QA_PLANS_PUB)

Specification APIs

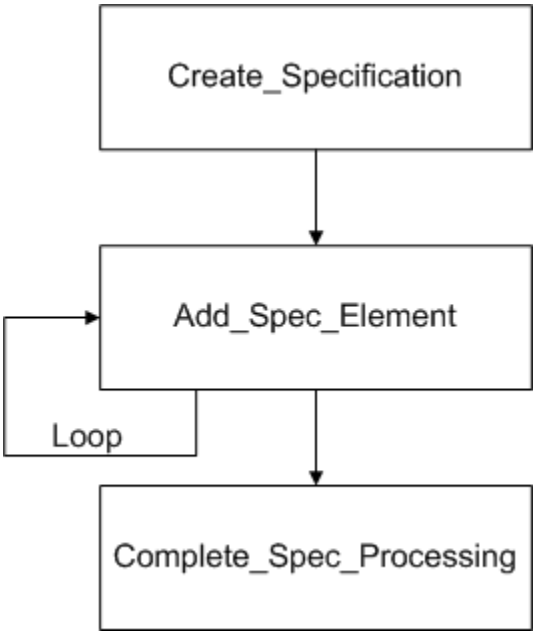
The APIs used to manipulate specification objects perform the following services:

- Create a specification
- Add a specification element to a specification
- Delete a specification
- Delete a specification element
- Copy a specification

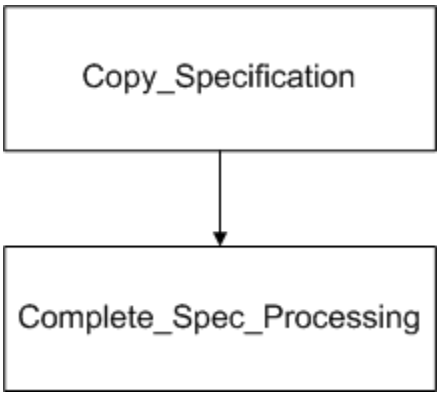
- Complete a specification definition

The following diagrams show how you can use the six specification APIs to perform four different tasks:

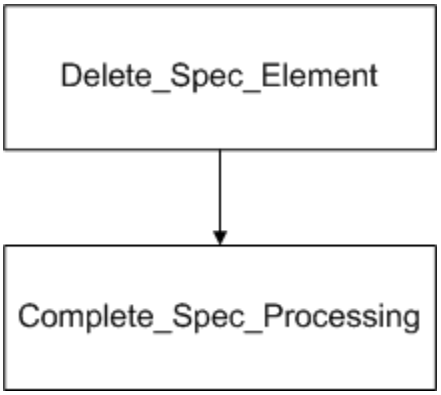
Create a Specification



Copy a Specification



Delete a Specification Element from an Existing Specification



Delete a Specification

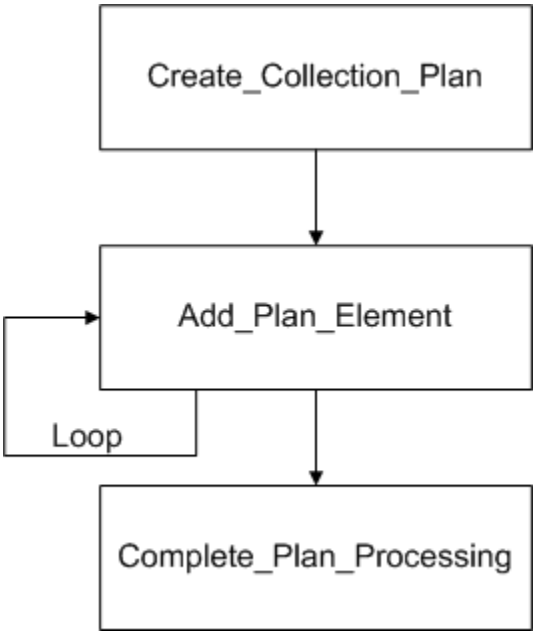
Collection Plan APIs

The APIs used to manipulate collection plan objects perform the following services:

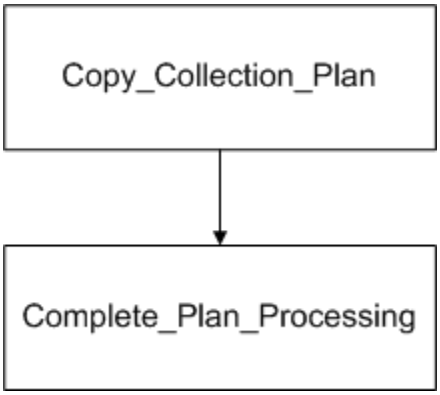
- Create a collection plan
- Add a collection plan element to a collection plan
- Delete a collection plan
- Delete a collection plan element
- Copy a collection plan
- Complete a collection plan definition

The following diagrams show how you can use the six collection plan APIs to perform four different tasks:

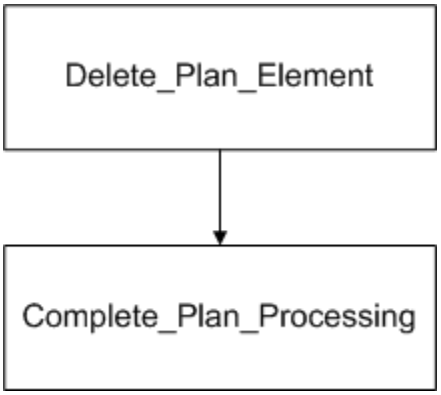
Create a Collection Plan



Copy a Collection Plan



Delete a Collection Element from an Existing Collection Plan



Delete a Collection Plan

Create Specification Application Program Interface

The Create Specification API, named create_specification, is a public API used to create a new specification header in Oracle Quality. After calling the create_specification procedure, call the Add Specification Element API (add_spec_element) consecutively to add as many specification elements as needed, then call the Complete Specification Processing API (complete_spec_processing) to finish the specification creation.

Setting Up the Create Specification API

The following describes how to setup the Create Specification API.

Parameter Descriptions

The following chart lists all parameters used by the Create Specification API. Additional information on these parameters follows the chart.

CREATE_SPECIFICATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			x
p_user_name	IN	Varchar2		x	
p_spec_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_reference_spec	IN	Varchar2			x
p_effective_from	IN	Date			x
p_effective_to	IN	Date			x
p_assignment_type	IN	Number			x
p_category_set_name	IN	Varchar2			x
p_category_name	IN	Varchar2			x
p_item_name	IN	Varchar2			x
p_item_revision	IN	Varchar2			x

Parameter	Usage	Type	Required	Derived	Optional
p_supplier_name	IN	Varchar2			x
p_customer_name	IN	Varchar2			x
p_sub_type_element	IN	Varchar2			x
p_sub_type_element_value	IN	Varchar2			x
x_spec_id	OUT	Number	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_spec_name

Specification name. Mixed case allowed.

p_organization_code

Organization code.

p_reference_spec

The parent specification, if the specification created is a child specification.

Default Value: NULL

p_effective_from

Effective From date.

Default Value: SYSDATE

p_effective_to

Effective To date.

Default Value: SYSDATE

p_assignment_type

Each specification has an assignment type. The assignment type indicates the association of a specification to either an item, a customer, or a supplier. The following values are allowed:

- 1 = Item Spec (qa_specs_pub.g_spec_type_item)
- 2 = Supplier Spec (qa_specs_pub.g_spec_type_supplier)
- 3 = Customer Spec (qa_specs_pub.g_spec_type_customer)

Default Value: QA_SPECS_PUB.G_SPEC_TYPE_ITEM

p_category_set_name

You can also associate a specification with a category and category set. This parameter specifies the category set and must be NULL if p_item_name is specified.

Default Value: NULL

p_category_name

You can also associate a specification with a category and category set. This parameter specifies the category and must be NULL if p_item_name is specified.

Default Value: NULL

p_item_name

The item name associated with this specification. This parameter must be NULL if p_category_set_name and p_category_name are specified.

Default Value: NULL

p_item_revision

The item revision associated with this specification. This parameter must be NULL if p_category_set_name and p_category_name are specified.

Default Value: NULL

p_supplier_name

The supplier associated with this specification.

Default Value: NULL

p_customer_name

The customer associated with this specification.

Default Value: NULL

p_sub_type_element

A specification can be tagged by a collection element its corresponding value. This indicates the collection element name.

Default Value: NULL

p_sub_type_element_value

A specification can be tagged by a collection element its corresponding value. This indicates the collection element value.

Default Value: NULL

x_spec_id

The specification ID created.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Create Specification API

The following describes the validation of Create Specification API.

Standard Validation

Oracle Quality validates all input parameters in the Create Specification API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Create Specification API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual .

Add Specification Element Application Program Interface

The Add Specification Element API, named add_spec_element, is a public API used to add a specification element to an existing specification in Oracle Quality. This procedure is most often used to add specification elements to specifications newly created by the Create Specification API (create_specification). Call the Add Specification Element API consecutively to add as many specification elements as needed, then call the Complete Specification Processing API (complete_spec_processing) to finish the specification creation.

Setting Up the Add Specification Element API

The following describes how to set up the Add Specification Element API.

Parameter Descriptions

The following chart lists all parameters used by the Add Specification Element API. Additional information on these parameters follows the chart.

ADD_SPEC_ELEMENT

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			x
p_user_name	IN	Varchar2		x	
p_spec_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_element_name	IN	Varchar2			x
p_uom_code	IN	Varchar2			x

Parameter	Usage	Type	Required	Derived	Optional
p_target_value	IN	Varchar2			x
p_enabled_flag	IN	Varchar2			x
p_upper_spec_limit	IN	Varchar2			x
p_lower_spec_limit	IN	Varchar2			x
p_upper_reasonable_limit	IN	Varchar2			x
p_lower_reasonable_limit	IN	Varchar2			x
p_upper_user_defined_limit	IN	Varchar2			x
p_lower_user_defined_limit	IN	Varchar2			x
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_spec_name

Specification name. Mixed case allowed.

p_organization_code

Organization code.

p_element_name

Name of the new specification element. Must be the name of an existing collection element.

p_uom_code

The unit of measure code chosen for this specification element.

Default Value: NULL

p_enabled_flag

Indicates whether this specification element is enabled. The following values are allowed:

- FND_API.G_TRUE
- FND_API.G_FALSE

Default Value: FND_API.G_TRUE

p_target_value

Specification target value.

Default Value: NULL

p_upper_spec_limit

Upper specification limit.

Default Value: NULL

p_lower_spec_limit

Lower specification limit.

Default Value: NULL

p_upper_reasonable_limit

Upper reasonable limit.

Default Value: NULL

p_lower_reasonable_limit

Lower reasonable limit.

Default Value: NULL

p_upper_user_defined_limit

Upper user defined limit.

Default Value: NULL

p_lower_user_defined_limit

Lower user defined limit.

Default Value: NULL

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Add Specification Element API

The following describes the validation of Add Specification Element API.

Standard Validation

Oracle Quality validates all input parameters in the Add Specification Element API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Add Specification Element API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Complete Specification Application Program Interface

The Complete Specification API, named complete_spec_processing, is a public API used to complete the definition of a new specification in Oracle Quality. This procedure is most often used to complete specifications newly created by the Create Specification (create_specification) and Add Specification Element (add_spec_element) APIs. Call the Complete Specification API after calling the Create Specification (create_specification) and Add Specification Element (add_spec_element) APIs.

Setting Up the Complete Specification API

The following describes how to setup the Complete Specification API.

Parameter Descriptions

The following chart lists all parameters used by the Complete Specification API. Additional information on these parameters follows the chart.

COMPLETE_SPEC_PROCESSING

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_user_name	IN	Varchar2		x	
p_spec_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_spec_name

Specification name. Mixed case allowed.

p_organization_code

Organization code.

p_commit

Indicates whether the API performs a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS

- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Complete Specification API

The following describes how to validate the Complete Specification API.

Standard Validation

Oracle Quality validates all input parameters in the Complete Specification API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Complete Specification API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual.

Delete Specification Application Program Interface

The Delete Specification API, named delete_specification, is a public API used to delete specifications in Oracle Quality. This procedure deletes the specification header and the associated specification elements.

Setting Up the Delete Specification API

The following describes the setting up of the Delete Specification API.

Parameter Descriptions

The following chart lists all parameters used by the Delete Specification API. Additional information on these parameters follows the chart.

DELETE_SPECIFICATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_user_name	IN	Varchar2		x	
p_spec_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F

- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_spec_name

Specification name. Mixed case allowed.

p_organization_code

Organization code.

p_commit

Indicates whether the API performs a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Delete Specification API

The following describe the validation of Delete Specification API.

Standard Validation

Oracle Quality validates all input parameters in the Delete Specification API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Delete Specification API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual.

Delete Specification Element Application Program Interface

The Delete Specification Element API, named delete_spec_element, is a public API used to delete specification elements in Oracle Quality. This procedure deletes a specification element from an existing specification.

Setting Up the Delete Specification Element API

The following describes the setting up of the Delete Specification Element API.

Parameter Descriptions

The following chart lists all parameters used by the Delete Specification Element API. Additional information on these parameters follows the chart.

DELETE_SPEC_ELEMENT

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_user_name	IN	Varchar2		x	
p_spec_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_element_name	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message

- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_spec_name

Specification name. Mixed case allowed.

p_organization_code

Organization code.

p_element_name

The name of the specification element to be deleted.

p_commit

Indicates whether the API performs a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR

- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Delete Specification Element API

The following describes the validation of the Delete Specification Element API.

Standard Validation

Oracle Quality validates all input parameters in the Delete Specification Element API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Delete Specification Element API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual.

Copy Specification Application Program Interface

The Copy Specification API, named copy_specification, is a public API used to copy specifications in Oracle Quality. This procedure copies an existing specification.

Setting Up the Copy Specification API

The following describes the setting up fo the Copy Specification API.

Parameter Descriptions

The following chart lists all parameters used by the Copy Specification API. Additional information on these parameters follows the chart.

COPY_SPECIFICATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_user_name	IN	Varchar2		x	
p_spec_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_to_spec_name	IN	Varchar2	x		
p_to_organization_code	IN	Varchar2	x		
p_to_item_name	IN	Varchar2	x		
p_to_item_revision	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_spec_id	OUT	Number	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_spec_name

Original specification name. Mixed case allowed.

p_organization_code

Organization code.

p_to_spec_name

Target Specification name. Mixed case allowed.

p_to_organization_code

Target Organization code.

p_to_item_name

The new item to be associated with the new specification.

p_to_item_revision

The new item revision. Null allowed.

Default Value: NULL

p_commit

Indicates whether the API performs a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

x_spec_id

Specification ID of the new specification.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Copy Specification API

The following describes the validation of Copy Specification API.

Standard Validation

Oracle Quality validates all input parameters in the Copy Specification API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Copy Specification API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual. This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Using the Specification APIs

Call the CREATE_SPECIFICATION subroutine. Ensure that the required fields (see: Create Specification Application Program Interface) are populated. If any validations fail, the user receives an error message. The user can also check the return status to verify the success or failure of the subroutine (see: Validation Of Create Specification API). If a row passes the validations, it is inserted in the QA_SPECS table, creating a new specification header record.

Next, call ADD_SPEC_ELEMENT (see: Add Specification Element Application Progeam Interface) to add a specification element to the specification header record. Again, validations occur. If a row passes the validations, it is inserted into the QA_SPEC_CHARS table. Call ADD_SPEC_ELEMENT as many times as necessary, then call COMPLETE_SPEC_PROCESSING (see: Complete Specification Application Program Interface). This subroutine uses the p_commit_flag value to commit changes to your database and avoid data integrity problems.

You can now call the following subroutines, followed by COMPLETE_SPEC_PROCESSING, to change your newly created specification, if necessary:

- DELETE_SPECIFICATION (see: Delete Specification Application Program Interface). It is not necessary to call COMPLETE_SPEC_PROCESSING afterwards.
- DELETE_SPEC_ELEMENT (see: Delete Specification Element Application Program Interface)
- COPY_SPECIFICATION (see: Copy Specification Application Program Interface)

Notes

If you provide both the item and the category set, the item takes precedence and is

inserted into the row.You cannot query a row added using CREATE_SPECIFICATION in Oracle Applications until after you have called ADD_SPEC_ELEMENT at least once and then called COMPLETE_SPEC_PROCESSING.

Specification APIs Example

```
DECLARE
--------------------------------------------------------------------------------
org_code                        VARCHAR2(3);
mesg_count                      NUMBER;
return_status                   VARCHAR2(500);
mesg_data                       VARCHAR2(500);
item                            VARCHAR2(150);
name                            VARCHAR2(30);
copy_spec                       VARCHAR2(30);
element                         VARCHAR2(30);
spec_id                         NUMBER;
effective_from                  DATE;
--------------------------------------------------------------------------------
BEGIN
org_code := 'M1';
name := 'Test Spec API';
item := 'CM82333';
effective_from := '01-FEB-2000';
```

Note: The next part of the sample code calls CREATE_SPECIFICATION to create an entry in the QA_SPECS table.

Note: p_api_version is a mandatory parameter. The current version is 1.0.

Note: p_init_msg_list is set to True to initialize the message stack.

Note: spec_id is the place holder for the specification identification number that is generated as a result of this call.

```
qa_specs_pub.create_specification(
```

p_api_version	=> 1.0,
p_init_msg_list	=> FND_API.G_TRUE,
p_spec_name	=> name,
p_organization_code	=> org_code,
p_effective_from	=> '01-FEB-1999',
p_item_name	=> item,
p_item_revision	=> 'B',
x_spec_id	=> spec_id,
x_return_status	=> return_status,
x_msg_count	=> mesg_count,
x_msg_data	=> mesg_data);

```
if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Spec Created: ' || name || ' ' || spec_id);
end if;
element := 'Quantity Defective';
qa_specs_pub.add_spec_element(
```

p_api_version	=> 1.0,
p_init_msg_list	=> FND_API.G_TRUE,
p_spec_name	=> name,
p_element_name	=> element,
p_target_value	=> 10,
x_return_status	=> return_status,
x_msg_count	=> mesg_count,
x_msg_data	=> mesg_data);

```
if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Spec Element Added: ' || element);
end if;
```

> **Note:** The next part of the sample code calls CREATE_SPEC_PROCESSING to perform the final validation. After final validation, p_commit_flag is set to TRUE if there are no errors. The row is now committed.

```
qa_specs_pub.complete_spec_processing(
```

p_api_version	=> 1.0
p_spec_name	=> name,
p_commit_flag	=> FND_API.G_TRUE,
x_return_status	=> return_status

```
x_msg_count                    => mesg_count,
x_msg_data                     => mesg_data);

if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Spec Definition Completed: ' || name);
end if;
end;
/
```

Create Collection Plan Application Program Interface

The Create Collection Plan API, named create_collection_plan, is a public API used to create new collection plans in Oracle Quality. This procedure creates the header of a new collection plan. After calling the create_collection_plan procedure, call the Add Collection Plan Element API (add_plan_element) consecutively to add as many collection plan elements as needed, then call the Complete Collection Plan Processing API (complete_plan_processing) to finish creating the collection plan.

Setting Up the Create Collection Plan API

The following describes the setting up of the Create Collection Plan API.

Parameter Descriptions

The following chart lists all parameters used by the Create Collection Plan API. Additional information on these parameters follows the chart.

CREATE_COLLECTION_PLAN

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		

Parameter	Usage	Type	Required	Derived	Optional
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			
p_user_name	IN	Varchar2		x	
p_plan_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_plan_type	IN	Varchar2	x		
p_description	IN	Varchar2	x		
p_effective_from	IN	Date	x		
p_effective_to	IN	Date	x		
p_spec_assignment_type	IN	Number	x		
x_plan_id	OUT	Number	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call

FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_plan_name

New collection plan name. Automatically converted to upper case.

p_organization_code

Organization code.

p_plan_type

Collection plan type as defined in the Collection Plan Type window, Meaning field. This is the complete name or translation of the code instead of the plan type code itself. The seeded plan type codes and descriptions include:

1 = WIP Inspection (qa_plans_pub.g_plan_type_wip_inspection)

2 = Receiving Inspection (qa_plans_pub.g_plan_type_rcv_inspection)

3 = FGI Inspection (qa_plans_pub.g_plan_type_fgi_inspection)

4 = Field Returns (qa_plans_pub.g_plan_type_field_returns)

p_description

Description of the collection plan.

Default Value: NULL

p_effective_from

Effective From date.

Default Value: SYSDATE

p_effective_to

Effective To date.

Default Value: NULL

p_spec_assignment_type

A plan can be associated with a specification assignment type. The assignment type indicates the association of a specification to either an item, a customer, or a supplier. The following values are allowed:

- 1 = Item Specification (qa_plans_pub.g_spec_type_item)
- 2 = Supplier Specification (qa_plans_pub.g_spec_type_supplier)
- 3 = Customer Spec (qa_plans_pub.g_spec_type_customer)
- 4 = No Specification (qa_plans_pub.g_spec_type_none)

Default Value: QA_PLANS_PUB.G_SPEC_TYPE_NONE

x_plan_id

The plan ID of the newly created collection plan.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Create Collection Plan API

The following describes the validation of Create Collection Plan API.

Standard Validation

Oracle Quality validates all input parameters in the Create Collection Plan API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Create Collection Plan API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual.

Add Collection Plan Element Application Program Interface

The Add Collection Plan Element API, named add_plan_element, is a public API used to add a new collection element to an existing collection plan in Oracle Quality. This procedure is most often used to add collection elements to collection plans newly created by the Create Collection Plan API (create_collection_plan). Call the Add Collection Plan Element API consecutively to add as many collection elements as needed, then call the Complete Collection Plan Processing API (complete_plan_processing) to finish creating the collection plan.

Setting Up the Add Collection Plan Element API

The following describes the setting up of the Add Collection Plan Element API.

Parameter Descriptions

The following chart lists all parameters used by the Add Collection Plan Element API. Additional information on these parameters follows the chart.

ADD_PLAN_ELEMENT

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			
p_user_name	IN	Varchar2		x	
p_plan_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_element_name	IN	Varchar2	x		
p_prompt_sequence	IN	Number			x
p_prompt	IN	Varchar2			x
p_default_value	IN	Varchar2			x
p_enabled_flag	IN	Varchar2			x
p_mandatory_flag	OUT	Varchar2			x
p_displayed_flag		Varchar2			x

Parameter	Usage	Type	Required	Derived	Optional
p_read_only_flag	IN	Varchar2			x
p_ss_poplist_flag	IN	Varchar2			x
p_information_flag	IN	Varchar2			x
p_result_column_name		Varchar2			x
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_plan_name

New collection plan name. Automatically converted to upper case.

p_organization_code

Organization code.

p_element_name

Name of the new collection element to add to the collection plan. Mixed case.

p_prompt_sequence

The integer prompt sequence. A NULL value assigns the element as the last in the sequence of elements.

Default Value: NULL

p_prompt

The prompt for the collection plan element to appear as a column heading when entering quality results. A G_INHERIT value uses the collection element's existing prompt.

Default Value: G_INHERIT

p_default_value

Default value of the collection plan element. A G_INHERIT value uses the collection element's existing default value.

Default Value: G_INHERIT

p_enabled_flag

Enabled flag. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_TRUE

p_mandatory_flag

Mandatory flag. Valid values include FND_API.G_TRUE or FND_API.G_FALSE and G_INHERIT. A G_INHERIT value uses the collection element's existing mandatory flag value.

Default Value: G_INHERIT

p_displayed_flag

Display flag. Specifies whether this element displays when entering quality results. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_TRUE

p_read_only_flag

Read only flag. Specify whether to display this element as non-editable and non-updateable. Valid values are FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: NULL, which has the same meaning as FND_API.G_FALSE.

p_ss_poplist_flag

Display this element's list of values as a poplist on Quality Workbench instead of the default LOV pick list for faster data entry. Valid values are FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: NULL, which has the same meaning as FND_API.G_FALSE.

p_information_flag

Information flag. Displays the value of this element in the information column in the Quality Workbench application. Valid values are FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: NULL, which has the same meaning as FND_API.G_FALSE.

p_result_column_name

This parameter is intended for use by experienced users who are familiar with the underlying Oracle Quality data model. This parameter suggests a database column in the qa_results table for use as storage. Most users should accept the default value of NULL and let the API select the database column.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS

- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Add Collection Plan Element API

The following describes the validation of Add Collection Plan Element.

Standard Validation

Oracle Quality validates all input parameters in the Add Collection Plan Element API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Add Collection Plan Element API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual

Complete Collection Plan Processing Application Program Interface

The Complete Collection Plan Processing API, named complete_plan_processing, is a public API used to complete the definition of a new collection plan in Oracle Quality. This procedure is most often used to complete collection plans newly created by the Create Collection Plan (create_collection_plan) and Add Collection Plan Element (add_plan_element) APIs. Call the Complete Collection Plan Processing API after calling the Create Collection Plan (create_collection_plan) and Add Collection Plan Element (add_plan_element) APIs.

Setting Up the Complete Collection Plan Processing API

The following describes the setting up of the Complete Collection Plan Processing API.

Parameter Descriptions

The following chart lists all parameters used by the Complete Collection Plan Processing API. Additional information on these parameters follows the chart.

COMPLETE_PLAN_PROCESSING

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			
p_user_name	IN	Varchar2		x	
p_plan_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is

greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_plan_name

New collection plan name. Automatically converted to upper case.

p_organization_code

Organization code.

p_commit

Indicates whether the API must perform a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

Oracle recommends performing a database commit, even though standards require the use of FND_API.G_FALSE as the default value. The dynamic plan views generator only executes if the user commits. Alternatively, launch the view generator manually in the Setup Collection Plans window.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the

actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Complete Collection Plan Processing API

The following describes the validation of the Complete Collection Plan Processing API.

Standard Validation

Oracle Quality validates all input parameters in the Complete Collection Plan Processing API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Complete Collection Plan Processing API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual .

Delete Collection Plan Application Program Interface

The Delete Collection Plan API, named delete_collection_plan, is a public API used to delete an existing collection plan and all of its collection elements in Oracle Quality. You

cannot delete a collection plan if quality results have been collected for the plan.

Setting Up the Delete Collection Plan API

The following describes the setting up of the Delete Collection Plan API.

Parameter Descriptions

The following chart lists all parameters used by the Delete Collection Plan API. Additional information on these parameters follows the chart.

DELETE_COLLECTION_PLAN

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			
p_user_name	IN	Varchar2		x	
p_plan_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_plan_name

New collection plan name. Automatically converted to upper case.

p_organization_code

Organization code.

p_commit

Indicates whether the API must perform a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

Oracle recommends performing a database commit, even though standards require the use of FND_API.G_FALSE as the default value. The dynamic plan views generator only executes if the user commits. Alternatively, launch the view generator manually in the Setup Collection Plans window.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Delete Collection Plan API

The following describes the validation of Delete Collection Plan API.

Standard Validation

Oracle Quality validates all input parameters in the Delete Collection Plan API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Delete Collection Plan API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Delete Collection Plan Element Application Program Interface

The Delete Collection Plan Element API, named delete_plan_element, is a public API used to delete an element from an existing collection plan in Oracle Quality. You cannot delete an element if a quality result has been collected for the element.

Setting Up the Delete Collection Plan Element API

The following describes the setting up of the Delete Collection Plan API

Parameter Descriptions

The following chart lists all parameters used by the Delete Collection Plan Element API. Additional information on these parameters follows the chart.

DELETE_PLAN_ELEMENT

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			
p_user_name	IN	Varchar2		x	
p_plan_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_element_name	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_plan_name

New collection plan name. Automatically converted to upper case.

p_organization_code

Organization code.

p_element_name

The element to delete. Mixed case.

p_commit

Indicates whether the API must perform a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

Oracle recommends performing a database commit, even though standards require the use of FND_API.G_FALSE as the default value. The dynamic plan views generator only executes if the user commits. Alternatively, launch the view generator manually in the Setup Collection Plans window.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Delete Collection Plan Element API

Standard Validation

Oracle Quality validates all input parameters in the Delete Collection Plan Element API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Delete Collection Plan Element API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Copy Collection Plan Application Program Interface

The Copy Collection Plan API, named copy_collection_plan, is a public API used in one of two ways:

- Copy an existing collection plan and all of its collection elements into a new collection plan
- Update an existing collection plan with additional collection elements copied from a source collection plan. If any of the new collection elements are in conflict with existing collection elements, the new elements from the source plan do not overwrite the existing elements.

After calling the copy_collection_plan procedure, call the Complete Collection Plan Processing API (complete_plan_processing) to finish the plan creation.

Setting Up the Copy Collection Plan API

The following describes the setting up of the Copy Collection Plan API.

Parameter Descriptions

The following chart lists all parameters used by the Copy Collection Plan API. Additional information on these parameters follows the chart.

COPY_COLLECTION_PLAN

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_validation_level	IN	Number			
p_user_name	IN	Varchar2		x	
p_plan_name	IN	Varchar2	x		
p_organization_code	IN	Varchar2	x		
p_to_plan_name	IN	Varchar2	x		
p_to_organization_code	IN	Varchar2	x		
p_copy_actions_flag	IN	Varchar2	x		
p_copy_values_flag	IN	Varchar2	x		
p_copy_transactions_flag	IN	Varchar2	x		
p_commit	IN	Varchar2	x		
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => l_message
- p_msg_index_out => l_msg_index_out

  where l_message and l_msg_index_out are local variables of types Varchar2 (2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

p_user_name

The user's name, as defined in the fnd_user table. This parameter is used to record audit info in the WHO columns.

Default Value: the user name derived from FND_GLOBAL.USER_ID

p_plan_name

New collection plan name. Automatically converted to upper case.

p_organization_code

Organization code.

p_to_plan_name

The destination plan name. Automatically converted to upper case.

p_to_organization_code

The destination organization code.

p_copy_actions_flag

Specifies whether to copy all actions associated with each collection plan element. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_TRUE

p_copy_values_flag

Specifies whether to copy all lookup values associated with each collection plan element. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_TRUE

p_copy_transactions_flag

Specifies whether to copy all transactions and collection triggers associated with each collection plan element. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_TRUE

p_commit

Indicates whether the API must perform a database commit. Valid values include FND_API.G_TRUE or FND_API.G_FALSE.

Default Value: FND_API.G_FALSE

Oracle recommends performing a database commit, even though standards require the use of FND_API.G_FALSE as the default value. The dynamic plan views generator only executes if the user commits. Alternatively, launch the view generator manually in the Setup Collection Plans window.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of Copy Collection Plan API

The following describes the validation of Copy Collection Plan API.

Standard Validation

Oracle Quality validates all input parameters in the Copy Collection Plan API. For specific information on the data implied by these parameters, see your Oracle Quality Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Copy Collection Plan API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

See Also

Oracle Applications Message Reference Manual.

Using the Collection Plan APIs

Call the CREATE_COLLECTION_PLAN subroutine. Ensure that the required fields (see: Create Collection Plan Application Program Interface) are populated. If any validations fail, the user receives an error message. The user can also check the return status to verify the success or failure of the subroutine (see: Validation of Create Collection Plan API). If a row passes the validations, it is inserted in the QA_PLANS table, creating a new collection plan header record.

Next, call ADD_PLAN_ELEMENT (see: Add Collection Plan Element Application Program Interface) to add a collection plan element to the collection plan header record. Again, validations occur. If a row passes the validations, it is inserted into the QA_PLAN_CHARS table. Call ADD_PLAN_ELEMENT as many times as necessary, then call COMPLETE_PLAN_PROCESSING (see: Complete Collection Plan Processing Application Program Interface). This subroutine uses the p_commit_flag value to commit changes to your database and avoid data integrity problems.

You can now call the following subroutines, followed by COMPLETE_PLAN_PROCESSING, to change your newly created collection plan, if necessary:

- DELETE_COLLECTION_PLAN (see: Delete Collection Plan Application Program Interface). It is not necessary to call COMPLETE_PLAN_PROCESSING afterwards.
- DELETE_PLAN_ELEMENT (see: Delete Collection Plan Element Application Program Interface)
- COPY_COLLECTION_PLAN (see: Copy Collection Plan Application Program Interface)

Note: You cannot query a row added using CREATE_COLLECTION_PLAN in Oracle Applications until after you have called ADD_PLAN_ELEMENT at least once and then called COMPLETE_PLAN_PROCESSING.

Collection Plan APIs Example

```
DECLARE
plan_id                 NUMBER;
org_code                VARCHAR2(3);
mesg_count              NUMBER;
return_status           VARCHAR2(500);
mesg_data               VARCHAR2(500);
description             VARCHAR2(150);
name                    VARCHAR2(30);
element_id              NUMBER;
element_name            VARCHAR2(30);
transaction_name        VARCHAR2(80);
action_name             VARCHAR2(80);
BEGIN
org_code := 'M1';
```

name := 'Plan API Test';

description := 'This Plan Is Created Using API';

> **Note:** The next part of the sample code calls CREATE_COLLECTION_PLAN to create an entry in the QA_PLANS table.
>
> p_api_version is a mandatory parameter. The current version is 1.0.
>
> p_init_msg_list is set to True to initialize the message stack.
>
> plan_id is the place holder for the collection plan identification number that is generated as a result of this call.

```
qa_plans_pub.create_collection_plan(
```

p_api_version	=> 1.0,
p_init_msg_list	=> FND_API.G_TRUE,
p_organization_code	=> org_code,
p_name	=> name,
p_plan_type	=> 'Test 11.5',
p_description	=> description,
x_plan_id	=> plan_id,
x_return_status	=> return_status,
x_msg_count	=> mesg_count,
x_msg_data	=> mesg_data);

```
if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Plan Created: ' || plan_id);
```

```
end if;
element_name := 'Quantity';
qa_plans_pub.add_plan_element(
```

p_api_version	=> 1.0,
p_plan_name	=> name,
p_element_name	=> element_name,
p_mandatory_flag	=> 'Y',
x_return_status	=> return_status,
x_msg_count	=> mesg_count,
x_msg_data	=> mesg_data);

```
if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Element Added: '|| element_name);
end if;
element_name := 'Defect Code';
qa_plans_pub.add_plan_element(
```

p_api_version	=> 1.0
p_spec_name	=> name,
p_element_name	=> element_name,
x_return_status	=> return_status
x_msg_count	=> mesg_count,

x_msg_data	=> mesg_data);

```
if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Element Added: '|| element_name);
end if;
```

> **Note:** The next part of the sample code calls CREATE_PLAN_PROCESSING to perform the final validation. After final validation, p_commit_flag is set to TRUE if there are no errors. The row is now committed.

```
qa_plans_pub.complete_plan_processing(
```

p_api_version	=> 1.0
p_spec_name	=> name,
p_commit_flag	=> FND_API.G_TRUE,
x_return_status	=> return_status
x_msg_count	=> mesg_count,
x_msg_data	=> mesg_data);

```
if return_status = FND_API.G_RET_STS_ERROR
or return_status = FND_API.G_RET_STS_UNEXP_ERROR then
fnd_msg_pub.get(fnd_msg_pub.g_first, fnd_api.g_false, mesg_data, mesg_count);
dbms_output.put_line(mesg_data);
else
dbms_output.put_line('Plan Definition Completed : '|| name);
end if;
```

```
end;
/
```

12

Oracle Work in Process Open Interfaces

This chapter covers the following topics:

- Open Move Transaction Interface
- Open Resource Transaction Interface
- Work Order Interface

Open Move Transaction Interface

You can load Move transaction information into the Open Move Transaction Interface from a variety of sources, including external data collection devices such as bar code readers, automated test equipment, cell controllers, and other manufacturing execution systems. You then use the interface to load these transactions into Oracle Work in Process. All transactions are validated and invalid transactions are marked, so that you can correct and resubmit them.

The Open Move Transaction Interface enables you to perform many of the functions possible from the Move Transactions window. For example, you can:

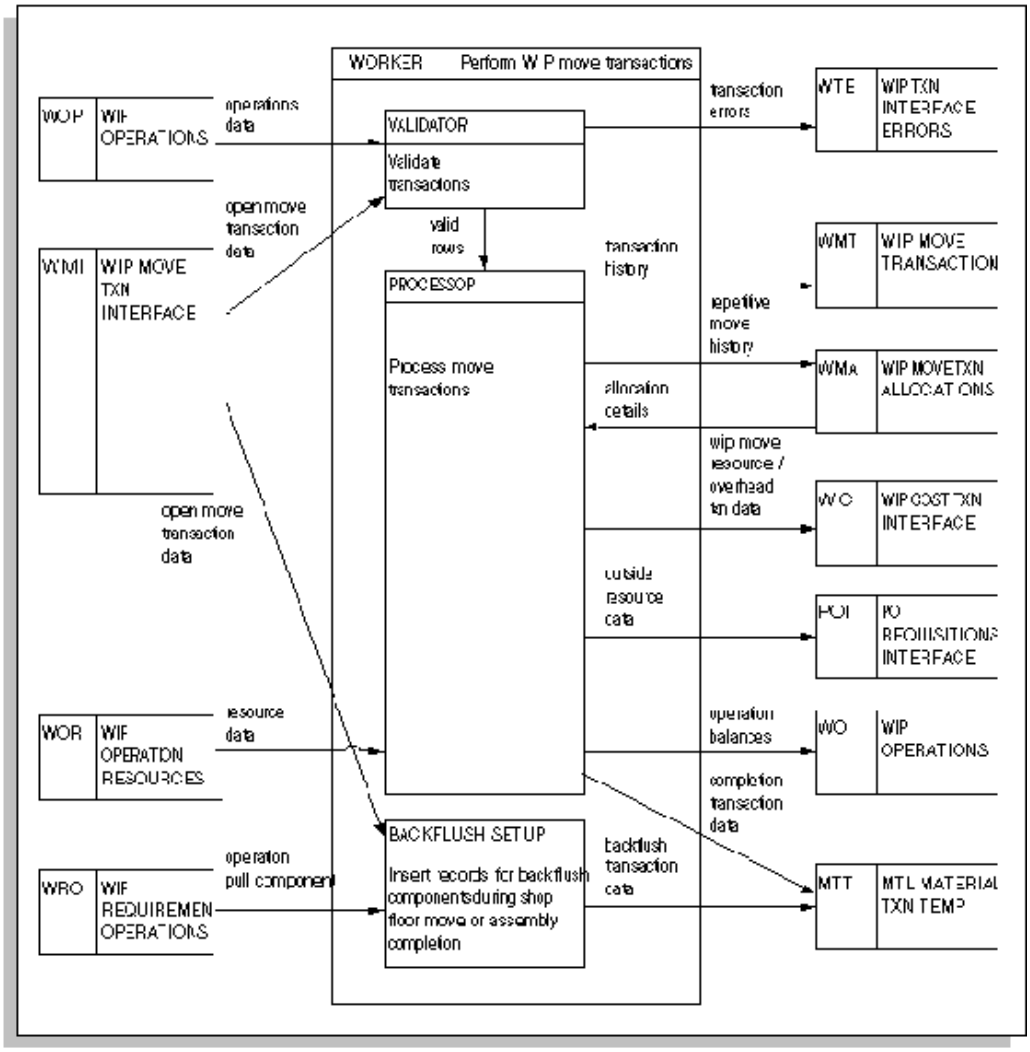
- Move assemblies between operations and intraoperation steps
- Scrap assemblies
- Move assemblies from an operation and complete them into inventory with a single transaction
- Over-complete a quantity greater than the job or schedule quantity if over-completions are enabled
- Return assemblies from inventory and move to an operation with a single transaction

Note: You cannot add ad hoc operations through this interface even

if the *WIP Allow Creation of New Operations* parameter is set.

Functional Overview

The following data flow diagram shows the key tables and programs that comprise the Move Transaction Interface:



You must write the load program that inserts a single row for each Move transaction into the WIP_MOVE_TXN_INTERFACE table. You must also insert records into the CST_COMP_SNAP_ INTERFACE table, if you insert Move transactions that also complete or return job assemblies, and if the referenced organization uses average costing. The system uses this information to calculate Completion cost. The Move Transaction Manager (WICTMS) then groups these transaction rows and launches a Move Transaction Worker to process each group.

The Move Transaction Worker calls the WIP Transaction Validation Engine program, which validates the row, derives or defaults any additional columns, and inserts errors into the WIP_TXN_INTERFACE_ERRORS table.

Next, the Move Transaction Processor performs the actual Move transaction. It writes it to history, allocates it to the correct Repetitive schedule (for Repetitive manufacturing only), initiates related resource and overhead transactions and requisitions for outside resources (for outside processing only), updates operation balances, initiates Completion transactions (for combination Move and Completion/Return transactions), and deletes successfully processed transaction rows from the WIP_MOVE_TXN_INTERFACE table. The Backflush Setup program then determines and initiates related operation pull backflushes.

If any transactions failed processing due to validation or other errors, you use the Pending Move Transactions window (WIPTSUPD) to review pending Move transactions and to update or delete failed transactions.

Completion Cost Detail Relationships

If the Move transaction also completes or returns job assemblies in an average costing organization, you need to link the Completion cost detail rows to their parent rows. You accomplish this by populating the WIP_MOVE_TXN_INTERFACE.TRANSACTION_ID with a unique value to be used as the primary key that links the child Completion cost rows. You must also populate the foreign key CST_COMP_SNAP_INTERFACE.TRANSACTION_INTERFACE_ID with the same value for all child Completion cost rows.

Setting Up the Move Transaction Interface

You must perform all the Oracle Bills of Material and Oracle Work in Process setup activities required for Move transactions. In addition, you must launch the Move Transaction Manager to process Move and combination Move and Completion/Return transactions that you import from external sources. See: Setting Up Shop Floor Control, *Oracle Work in Process User's Guide*

Launching the Move Transaction Manager

You launch the Move Transaction Manager in the Interface Managers window in Oracle Inventory. When you launch the Move Transaction Manager, you can specify the resubmit interval and number of transactions processed by each worker during each interval. After polling the WIP_MOVE_TXN_INTERFACE table for eligible rows, the Move Transaction Manager creates the necessary number of Move Transaction Workers to process the load.

The use of multiple transaction workers enables parallel processing of transactions that can be especially helpful when importing a large batch of transactions through the Move Transaction Interface. For more information, see: Transaction Managers, *Oracle Inventory User's Guide.*

Inserting Records into the WIP_MOVE_TXN_INTERFACE Table

You must insert your Move, Move Complete and Move Return transactions into the WIP_MOVE_TXN_INTERFACE table. The system validates each transaction row, derives any additional data as necessary, then processes each transaction.

WIP_MOVE_TXN_INTERFACE Table Description

Type Number	Transaction Type Description
1	Move transaction
2	Move Completion (defined by TRANSACTION_TYPE = 2 and valid FM_OPERATION_SEQ_NUM)
3	Move Return (defined by TRANSACTION_TYPE = 3 and valid T0_OPERATION_SEQ_NUM)

Column Name	Type	Required	Derived	Optional	Derived if Null
ACCT_PERIOD_ID	Number	-	1,2,3	-	-
ATTRIBUTE1-ATTRIBUTE15	Varchar2(150)	-	-	1,2,3	-
ATTRIBUTE_CATEGORY	Varchar2(30)	-	-	1,2,3	-
CREATED_BY	Number	-	1,2,3	-	-
CREATED_BY_NAME	VarChar2(100)	1,2,3	-	-	-
CREATION_DATE	Date	1,2,3	-	-	-

Column Name	Type	Required	Derived	Optional	Derived if Null
EMPLOYEE_ID	Number	Required only if MES parameter Required Badge for Move Transaction is set to Yes	-	1,2,3	-
ENTITY_TYPE	Number	-	1,2,3	-	-
FM_DEPARTMENT_CODE	VarChar2(10)	-	1,2	-	-
FM_DEPARTMENT_ID	Number	-	1,2	-	-
FM_INTRAOPERATION_STEP_TYPE	Number	1,2	-	-	-
FM_OPERATION_CODE	VarChar2(4)	-	1,2	-	-
FM_OPERATION_SEQ_NUM	Number	1,2	-	-	-
GROUP_ID	Number	-	1,2,3	-	-
LAST_UPDATE_DATE	Date	1,2,3	-	-	-
LAST_UPDATE_LOGIN	Number	-	1,2,3	-	-
LAST_UPDATED_BY	Number	-	1,2,3	-	-
LAST_UPDATED_BY_NAME	VarChar2(100)	1,2,3	-	-	-

Column Name	Type	Required	Derived	Optional	Derived if Null
LINE_CODE	VarChar2(10)	1,2,3	-	-	-
LINE_ID	Number	-	-	-	1,2,3
ORGANIZATION_CODE	VarChar2(3)	1,2,3	-	-	-
ORGANIZATION_ID	Number	-	-	-	1,2,3
OVERCOMPLETION_PRIMARY_QTY	Number	-	1,2,3	-	-
OVERCOMPLETION_TRANSACTION_ID	Number	-	1,2,3	-	-
OVERCOMPLETION_TRANSACTION_QTY	Number	-	1,2,3	-	-
PRIMARY_ITEM_ID	Number	-	1,2,3	-	-
PRIMARY_QUANTITY	Number	-	1,2,3	-	-
PRIMARY_UOM	VarChar2(3)	-	1,2,3	-	-
PROCESS_PHASE	Number	1,2,3	-	-	-
PROCESS_STATUS	Number	1,2,3	-	-	-
PROGRAM_APPLICATION_ID	Number	-	1,2,3	-	-
PROGRAM_ID	Number	-	1,2,3	-	-

Column Name	Type	Required	Derived	Optional	Derived if Null
PROGRAM_UPDATE_DATE	Date	-	1,2,3	-	-
QA_COLLECTION_ID	Number	-	-	1,2,3	-
REASON_ID	Number	-	-	-	1,2,3
REASON_NAME	VarChar2(30)	-	-	1,2,3	-
REFERENCE	VarChar2(240)	-	-	1,2,3	-
REPETITIVE_SCHEDULE_ID	Number	-	1,2,3	-	-
REQUEST_ID	Number	-	1,2,3	-	-
SCRAP_ACCOUNT_ID	Number	-	-	1,2,3	-
SOURCE_CODE	Varchar2(30)	-	-	1,2,3	-
SOURCE_LINE_ID	Number	-	-	1,2,3	-
TO_DEPARTMENT_CODE	VarChar2(10)	-	1,3	-	-
TO_DEPARTMENT_ID	Number	-	1,3	-	-
TO_INTRAOPERATION_STEP_TYPE	Number	1,3	-	-	-
TO_OPERATION_CODE	VarChar2(4)	-	1,3	-	-
TO_OPERATION_SEQ_NUM	Number	1,3	-	-	-

Column Name	Type	Required	Derived	Optional	Derived if Null
TRANSACTION_DATE	Date	1,2,3	-	-	-
TRANSACTION_ID	Number	-	1,2,3	-	-
TRANSACTION_QUANTITY	Number	1,2,3	-	-	-
TRANSACTION_TYPE	Number	-	-	1,2,3	-
TRANSACTION_UOM	VarChar2(3)	1,2,3	-	-	-
WIP_ENTITY_ID	Number	-	-	-	1,2,3
WIP_ENTITY_NAME	VarChar2(240)	1,2,3	-	-	-

You must include data in each of the required columns. Overall, very few columns are required because the system derives or defaults many column values and/or allows these column values to be optional.

Columns are derived using foreign key relationships within Oracle Manufacturing. The following derived columns are control columns that the Move Transaction Worker uses to provide closed loop transaction processing control and relational integrity throughout the interface process:

Control Columns

- ATTRIBUTE1 through ATTRIBUTE15 (Optional): the descriptive flexfield attributes in the columns ATTRIBUTE1 through ATTRIBUTE15 map to ATTRIBUTE1 through ATTRIBUTE15 in WIP_MOVE_TRANSACTIONS.

- FM_INTRAOPERATION_STEP_TYPE (Required): this column is only required when performing Move and Move Completion transactions. It must be an enabled intraoperation step.

- FM_OPERATION_SEQUENCE (Required): in *Move transactions*, this column represents the operation from which you are moving the assemblies.

  In *Move and Completion transactions,* this column represents the operation from which you are moving the assemblies before they are completed into inventory.

In *Move and Return transactions,* you may leave this column and the FM_INTRAOPERATION_STEP_TYPE column blank. If you do not wish to leave these columns blank when performing a Move and Return transaction, however, you must set the values of this column and the FM_INTRAOPERATION_STEP_TYPE column to their derived values: FM_OPERATION_SEQ_NUM must be set to the last operation sequence on the routing, and FM_INTRAOPERATION_STEP_TYPE must be set to To Move.

- LINE_CODE (Derived): this column is only required for Repetitive manufacturing transactions. If the WIP_ENTITY specified is a Repetitive assembly, the column is derived.

- ORGANIZATION_CODE (Derived): this column is used to derive the Organization ID. The Organization ID identifies the organization to which the transaction belongs.

- OVERCOMPLETION_PRIMARY_QTY (Derived): the OVERCOMPLETION_PRIMARY_QUANTITY is derived from the OVERCOMPLETION_TRANSACTION _QTY and OVERCOMPLETION_TRANSACTION_UOM.

- OVERCOMPLETION_TRANSACTION_QTY (Conditionally Required): if you intend to move and eventually complete more assemblies than exist at a routing operation, the OVERCOMPLETION_TRANSACTION_QTY is required. It cannot be derived.

- PRIMARY_QUANTITY (Derived): this column is the transaction quantity in the assembly's primary unit of measure, calculated using TRANSACTION_QUANTITY and TRANSACTION_UOM.

- PROCESS_PHASE (Required): the column PROCESS_PHASE describes the current processing phase of the transaction. The Move Transaction Worker processes each transaction row through the following three phases:

  - Move Validation

  - Move Processing

  - Operation Backflush Setup

  You should always load 1 (Move Validation)

- PROCESS_STATUS (Required): this column control describes the transaction state of the row and controls whether rows in the interface table are processed. You should insert a row that you intend to be processed with a value of 1 (Yes).

  - Pending

- Running
- Error

You should always load 1 (Pending)

- SCRAP_ACCOUNT_ID (Optional): if the TO_INTRAOPERATION_STEP_TYPE is scrap and a scrap account is required, you must insert a SCRAP_ACCOUNT_ID.
- SOURCE_CODE and SOURCE_LINE_ID (Optional): the SOURCE_CODE and SOURCE_LINE_ID columns can be used to identify the source of your Move transactions. For example, if you collect Move transaction information from a bar code reader and a radio frequency device, you could use a different source code to identify each collection method.
- TO_INTRAOPERATION_STEP_TYPE (Required): if you leave TO_INTRAOPERATION_STEP_TYPE blank, it is derived the same way as the TO_OPERATION_SEQ_NUM column. If you are moving and returning assemblies, you cannot specify the To Move intraoperation step of the last operation. If you specify the Scrap intraoperation step, you must insert a SCRAP_ACCOUNT_ID if the WIP *Require Scrap Account* parameter is set.
- TO_OPERATION_SEQ_NUM (Required): in *Move transactions*, this column represents the operation step into which you are moving the assemblies. If you are moving assemblies and leave this column and the TO_INTRAOPERATION_STEP_TYPE columns blank, both columns are derived. The TO_OPERATION_SEQ_NUM is set to the next count point operation and the TO_INTRAOPERATION_STEP_TYPE is set to Queue.

  In *Move and Return transactions,* this column represents the operation that the assemblies are being returned to from inventory. If you are moving and returning assemblies and leave this column and the TO_INTRAOPERATION_STEP_TYPE columns blank, both columns are also derived; however, the TO_OPERATION_SEQ_NUM is set to the last count point operation. If the last count point operation is the last operation sequence, the TO_OPERATION_SEQ_NUM column is set to the value of the operation prior to the last count point operation. Regardless, if there is no count point operation, and the TO_OPERATION_SEQ_NUM is blank, the transaction will fail validation.

  In *Move and Completion transactions,*you may leave this column and the TO_INTRAOPERATION_STEP_TYPE column blank. If you do not wish to leave them blank when performing a move and completion transaction, however, you must set the values of this column and the TO_INTRAOPERATION_STEP_TYPE column to their derived values: TO_OPERATION_SEQ_NUM must be set to the last operation sequence on the routing, and TO_INTRAOPERATION_STEP_TYPE must be set to To Move.
- TRANSACTION_QUANTITY (Required): enter the transaction quantity in the

same unit of measure used in the transaction. The quantity should be positive for receipts into inventory, and negative both for issues out of inventory and for transfers. Enter a quantity of zero for Average Cost Update transactions.

- TRANSACTION_TYPE (Optional): This column indicates the type of Move transaction. The options are:
  - Move
  - Move Completion
  - Move Return

  If you leave this column blank (NULL) the transaction is processed like a Move transaction. When TRANSACTION_TYPE is 2, the Move transaction processor moves the assemblies to the last operation in the routing and completes the units into the Completion subinventory/locator. When TRANSACTION_TYPE is 3, the Move transaction processor returns the units from the Completion subinventory/locator, and moves the assemblies to the last operation.

- TRANSACTION_UOM (Required): you can enter the TRANSACTION_QUANTITY in any unit of measure that has conversion rates defined for the items primary unit of measure. Use this column to specify the transacted unit of measure, even if it is the same as the primary unit of measure.

- WIP_ENTITY_NAME (Required): this column represents the job name or line/assembly association used to derive the WIP Entity ID.

Required Columns

- For normal *Move transactions,* set TRANSACTION_TYPE to 1 or NULL. If you leave TO_OPERATION_SEQ_NUM and TO_INTRAOPERATION_STEP_TYPE blank, the data are defaulted. The TO_OPERATION_SEQ_NUM is defaulted to the next count point operation in the routing, and the TO_INTRAOPERATION_STEP_TYPE is defaulted to Queue. If there is no count point operation and the TO_OPERATION_SEQ_NUM is blank, then the transaction fails validation.

- For combination *Move and Completion transactions*, set TRANSACTION_TYPE to 2. When TRANSACTION_TYPE is 2, the Move transaction processor moves the assemblies to the last operation in the routing and completes the units into the Completion subinventory/locator.

- For combination *Move and Return transactions,* set TRANSACTION_TYPE to 3. When TRANSACTION_TYPE is 3, the Move transaction processor returns the assemblies from the Completion subinventory/locator.

- The column LINE_CODE is required for Repetitive manufacturing transactions only.

- The column PROCESS_PHASE describes the current processing phase of the transaction. The Move Transaction Worker processes each transaction row through the following three phases. You should always load 1 (Move Validation). The options are:
  - Move Validation
  - Move Processing
  - Operation Backflush Setup
- The column PROCESS_STATUS contains the state of the transaction. You should always load 1 (Pending):
  - Pending
  - Running
  - Error

Derived Data

The WIP Transaction Validation Engine program derives columns using foreign key relationships within Oracle Manufacturing. The following derived columns are control columns that the Move Transaction Worker uses to provide closed loop transaction processing control and relational integrity throughout the interface process:

- CREATED_BY
- GROUP_ID
- LAST_UPDATE_LOGIN
- LAST_UPDATED_BY
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE
- REQUEST_ID
- TRANSACTION_ID

You can insert data into certain derived columns. The WIP Transaction Validation Engine will validate your data, but not override it. You can insert data into the following derived columns:

- LINE_ID

- ORGANIZATION_ID
- REASON_ID
- WIP_ENTITY_ID

Optional Columns

- The columns SOURCE_CODE and SOURCE_LINE_ID can be used to identify the source of Move transactions. For example, if you collect Move transaction information from a bar code reader and a radio frequency device, you could use a different source code to identify each collection method.
- The descriptive flexfield attributes in the columns ATTRIBUTE1 through ATTRIBUTE15 map to ATTRIBUTE1 through ATTRIBUTE15 in WIP_MOVE_TRANSACTIONS.

CST_COMP_SNAP_INTERFACE Table

The following table describes the CST_COMP_SNAP_ INTERFACE Interface:

Column Name	Type	Required	Derived	Optional
CREATED_BY	Number	x	-	-
CREATION_DATE	Date	x	-	-
LAST_UPDATE_DATE	Date	x	-	-
LAST_UPDATE_LOGIN	Number	-	-	x
LAST_UPDATED_BY	Number	x	-	-
NEW_OPERATION_FLAG	Number	-	x	-
OPERATION_SEQ_NUMBER	Number	x	-	-
PRIMARY_QUANTITY	Number	-	x	-
PRIOR_COMPLETION_QUANTITY	Number	-	x	-
PRIOR_SCRAP_QUANTITY	Number	-	x	-

Column Name	Type	Required	Derived	Optional
PROGRAM_APPLICATION_ID	Number	-	-	x
PROGRAM_ID	Number	-	-	x
PROGRAM_UPDATE_DATE	Date	-	-	x
QUANTITY_COMPLETED	Number	x	-	-
REQUEST_ID	Number	-	-	x
TRANSACTION_INTERFACE_ ID	Number	x	-	-
WIP_ENTITY_ID	Number	x	-	-

- NEW_OPERATION_FLAG: indicates whether or not the operation was added to the job after it was released.
- OPERATION_SEQ_NUMBER: you can use this column to enter operation sequence information. The operation sequence number is stored in the WIP_OPERATIONS table. The value that you enter here must agree with the value in the WIP_OPERATIONS table for this job.
- PRIMARY_QUANTITY: you can use this column to indicate the number of assemblies being completed or returned during a Move transaction.
- QUANTITY_COMPLETED: indicates the number of assemblies completed at or returned to the specified operation.
- WIP_ENTITY_ID: the WIP_ENTITY_ID is the job number.

Validating Move Transactions

The Move Transaction Manager program groups the Move transaction rows in the WIP_MOVE_TXN_INTERFACE and launches Move Transaction Workers to process each group. The Move Transaction Worker program calls the WIP Transaction Validation Engine program to validate and derive data for each of the required columns. If data are entered in certain derived and optional columns, the WIP Transaction Validation Engine program also validates these columns.

Before information is copied into CST_COMP_SNAP_TEMP, the information in the CST_COMP_SNAP_INTERFACE table is validated. The TRANSACTION_INTERFACE_ID is validated against the TRANSACTION_INTERFACE_ID in the Move Transaction Interface. The WIP_ENTITY_ID must exist in WIP_OPERATIONS, and the OPERATION_SEQ_NUM must match the OPERATION_SEQ_NUM in WIP_OPERATIONS for the specified WIP_ENTITY_ID (job).

The system considers the dependencies among the columns in the interface table and only processes columns once they are dependent upon pass validation or are successfully derived. For example, the Move validator only validates WIP_ENTITY_NAME after ORGANIZATION_ID has been derived. In turn, ORGANIZATION_ID is only derived after ORGANIZATION_CODE has been successfully validated.

The system creates rows in the WIP_TXN_INTERFACE_ERRORS table for each failed validation. Each row in the WIP_TXN_INTERFACE_ERRORS table contains the TRANSACTION_ID of the failed Move transaction, the name of the column that failed validation, and a brief error message stating the cause of the validation failure. Because of the dependencies between columns, the Move validator does not try to validate a column when a column it is dependent upon fails validation. So WIP_ENTITY_NAME is not validated if ORGANIZATION_CODE is invalid.

Columns that are independent of each other, however, can be validated regardless of the status of other columns. For example, WIP_ENTITY_NAME is validated even if REASON_NAME is invalid because WIP_ENTITY_NAME is not dependent on REASON_NAME. Thus, the system can create multiple error records for each Move transaction. You can then resolve multiple problems at the same time, thereby increasing the speed and efficiency of your error resolution process.

Resolving Failed Rows

Viewing Failed Rows

You view both pending and failed Move transaction rows in the WIP_MOVE_TXN_INTERFACE table, using the Pending Move Transactions window. You also can view errors associated with failed transactions by navigating to the Pending Move Transaction Errors window. See: Processing Pending Move Transactions, *Oracle Work in Process User's Guide*.

Fixing Failed Rows

You update failed Move transaction rows in the WIP_MOVE_TXN_INTERFACE table using the Pending Move Transactions window. Once you have made the necessary changes, you place a check mark in the Resubmit check box and save your work. The transaction row is then eligible to be picked up by the Move Transaction Manager for revalidation and processing. You also can have the system check the Resubmit check box for all queried rows, by selecting Select All for Resubmit from the windows *Tools Menu*. When you save your work, all of these rows become eligible for revalidation and processing. You also can use the Pending Move Transactions window to delete problem

rows from the WIP_MOVE_TXN_INTERFACE table. Deleting problem rows ensures that you do not have duplicate data when you reload the corrected data from the source application.

You can re-query transactions that fail an initial validation, then resubmit them after correcting the cause of the failure. For example, if the Move transaction failed because the job status was Unreleased, you can use the Discrete Jobs window to release the job then resubmit the pending Move transaction.

The Move processor creates resource cost transactions that are processed in the background by the Cost Manager. You can also update or resubmit these transactions using the Pending Move Transactions window. See: Processing Pending Move Transactions, *Oracle Work in Process User's Guide*

Open Resource Transaction Interface

You can use external data collection devices such as bar code readers, payroll systems, and time card entry forms to collect resource and overhead transaction data, then load the data into the Open Resource Transaction Interface for Oracle Work in Process to process. The interface validates the data and marks any that are invalid so that you can correct and resubmit them.

You can use this interface to perform many of the same functions that you can perform from the Resource Transactions window. For example, you can:

- Charge labor resources
- Charge overhead resources
- Charge outside processing resources
- Add ad hoc resources

> **Note:** Non-person resources cannot be charged at an actual usage rate or amount through the Resource Transactions window. This feature is unique to the Resource Transaction Interface.

Functional Overview

You must write the load program that inserts a single row for each resource transaction into the WIP_COST_TXN_INTERFACE table. The Cost Manager (CMCCTM) then groups these transaction rows and launches a Cost Worker to process each group.

The Cost Worker calls the WIP Transaction Validation Engine program which validates the row, derives or defaults any additional columns, and inserts errors into the WIP_TXN_INTERFACE_ERRORS table. The Cost Worker then performs the actual resource transaction, writing the transaction to history. It allocates Repetitive resource

transactions to the correct Repetitive schedules, updates operation resource balances, and deletes the successfully processed transaction row from the WIP_COST_TXN_INTERFACE table.

You then use the Pending Resource Transactions window to review any pending transactions and to update or delete transactions that failed processing due to validation or other errors.

Setting Up the Resource Transaction Interface

You must perform all of the Oracle Bills of Materials, Costing, and Work in Process setup activities required for resource and overhead transactions. In addition, you must launch the Cost Manager to process resource transactions that you import from external sources. See: Setting Up Resource Management, *Oracle Work in Process User's Guide*

Launching the Cost Manager

You launch the Cost Manager in Oracle Inventory's Interface Managers window. When you launch the Cost Manager, you specify the resubmit interval and the number of transactions processed by each worker during each interval. After polling the WIP_COST_TXN_INTERFACE table for eligible rows, the Cost Manager creates the necessary number of Cost Workers to process the load. The use of multiple transaction workers enables parallel transaction processing, which is especially helpful when processing a large batch of transactions imported through the Resource Transaction Interface. See: Transaction Managers, *Oracle Inventory User's Guide.*

Inserting Records into the WIP_COST_TXN_INTERFACE Table

You must insert your resource transactions into the WIP_COST_TXN_INTERFACE table. The system validates each transaction row, derives any additional data as necessary, then processes each transaction.

WIP_COST_TXN_INTERFACE Table Description

The following table describes the WIP_COST_TXN_INTERFACE:

Column Name	Type	Required	Derived	Optional
ACCT_PERIOD_ID	Number	-	x	-
ACTIVITY_ID	Number	-	x	x
ACTIVITY_NAME	VarChar2(10)	-	-	x

Column Name	Type	Required	Derived	Optional
ACTUAL_RESOURCE_RATE	Number	-	x	-
ATTRIBUTE1-ATTRIBUTE15	Varchar2(150)	-	-	x
ATTRIBUTE_CATEGORY	Varchar2(30)	-	-	x
AUTOCHARGE_TYPE	Number	-	x	-
BASIS_TYPE	Number	-	x	-
CREATED_BY	Number	-	x	-
CREATED_BY_NAME	VarChar2(100)	x	-	-
CREATION_DATE	Date	x	-	-
CURRENCY_ACTUAL_RESOURCE_RATE	Number	-	-	x
CURRENCY_CODE	VarChar2(15)	-	-	x
CURRENCY_CONVERSION_DATE	Date	-	x	x
CURRENCY_CONVERSION_RATE	Number	-	-	x
CURRENCY_CONVERSION_TYPE	VarChar2(10)	-	-	x
DEPARTMENT_CODE	VarChar2(10)	-	x	-
DEPARTMENT_ID	Number	-	x	-

Column Name	Type	Required	Derived	Optional
DISTRIBUTION_ACCOUNT_ID	Number	-	-	x
EMPLOYEE_ID	Number	-	x	x
EMPLOYEE_NUM	VarChar2(30)	-	-	x
ENTITY_TYPE	Number	-	x	-
GROUP_ID	Number	x	-	-
LAST_UPDATE_DATE	Date	x	-	-
LAST_UPDATE_LOGIN	Number	-	x	-
LAST_UPDATED_BY	Number	-	x	-
LAST_UPDATED_BY_NAME	VarChar2(100)	x	-	-
LINE_ID	Number	-	x	-
LINE_CODE	VarChar2(10)	-	-	x
MOVE_TRANSACTION_ID	Number	-	-	x
OPERATION_SEQ_NUM	Number	x	-	-
ORGANIZATION_CODE	VarChar2(3)	x	-	-
ORGANIZATION_ID	Number	x	-	-
PO_HEADER_ID	Number	-	-	x

Column Name	Type	Required	Derived	Optional
PO_LINE_ID	Number	-	-	x
PRIMARY_ITEM_ID	Number	-	x	-
PRIMARY_QUANTITY	Number	-	x	-
PRIMARY_UOM	VarChar2(3)	-	x	-
PRIMARY_UOM_CLASS	VarChar2(10)	-	x	-
PROCESS_PHASE	Number	x	-	-
PROCESS_STATUS	Number	x	-	-
PROGRAM_APPLICATION_ID	Number	-	x	-
PROGRAM_ID	Number	-	x	-
PROGRAM_UPDATE_DATE	Date	-	x	-
RCV_TRANSACTION_ID	Number	-	-	x
REASON_ID	Number	-	x	x
REASON_NAME	VarChar2(30)	-	-	x
RECEIVING_ACCOUNT_ID	Number	-	-	x
REFERENCE	VarChar2(240)	-	-	x
REPETITIVE_SCHEDULE_ID	Number	-	x	-

Column Name	Type	Required	Derived	Optional
REQUEST_ID	Number	-	x	-
RESOURCE_CODE	VarChar2(10)	-	x	x
RESOURCE_ID	Number	-	x	-
RESOURCE_SEQ_NUM	Number	x	-	-
RESOURCE_TYPE	Number	-	x	-
SOURCE_CODE	Varchar2(30)	-	-	x
SOURCE_LINE_ID	Number	-	-	x
STANDARD_RATE_FLAG	Number	-	x	-
TRANSACTION_DATE	Date	x	-	-
TRANSACTION_ID	Number	-	x	-
TRANSACTION_QUANTITY	Number	x	-	-
TRANSACTION_TYPE	Number	x	-	-
TRANSACTION_UOM	VarChar2(3)	x	-	-
USAGE_RATE_OR_AMOUNT	Number	-	x	-
WIP_ENTITY_ID	Number	-	x	x
WIP_ENTITY_NAME	VarChar2(240)	x	-	-

Note: You cannot load resource and overhead cost transactions for Flow schedules.

You must include data in each of the required columns. Overall, very few columns are required because the system derives or defaults many column values and/or allows these column values to be optional.

Note: Do not put a value in the COMPLETION_TXN_ID column.

Required Columns

- The column LINE_CODE is required for Repetitive manufacturing transactions only.
- The column PROCESS_PHASE describes the current processing phase of the transaction. The Cost Worker processes each transaction row through the following two phases (you should always load 1 Resource Validation).
  - Resource Validation
  - Resource Processing
- The column PROCESS_STATUS contains the state of the transaction. You should always load 1 (Pending):
  - Pending
  - Running
  - Error
- The column RESOURCE_ID column must be left NULL.
- You set TRANSACTION_TYPE to:
  - normal resource transactions
  - overhead transactions
  - outside processing transactions

Derived Data

The WIP Transaction Validation Engine program uses foreign key relationships within Oracle Manufacturing to obtain the values for the Derived columns in the above table.

The following Derived columns are control columns that the Cost Worker uses to provide closed loop transaction processing control and relational integrity throughout the interface process:

- CREATED_BY
- GROUP_ID
- LAST_UPDATE_LOGIN
- LAST_UPDATED_BY
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE
- REQUEST_ID
- TRANSACTION_ID

You have the option to insert data into certain derived columns. The WIP Transaction Validation Engine validates the data, but does not override them. You can insert data into the following derived columns:

- ACTIVITY_ID
- EMPLOYEE_ID
- LINE_ID
- REASON_ID
- WIP_ENTITY_ID

Optional Columns

- The descriptive flexfield attributes in the columns ATTRIBUTE1 through ATTRIBUTE15 map to ATTRIBUTE1 through ATTRIBUTE15 in WIP_TRANSACTIONS.
- You can use the columns SOURCE_CODE and SOURCE_LINE_ID to identify the source of your resource and overhead transactions. For example, if you collect resource data from a bar code reader and a labor data entry form, you can use a different source code to identify each collection method.

Costing Option

You can charge non-person resources (resources for which an EMPLOYEE_NUMBER is not specified) at their actual rate by specifying the rate in the USAGE_RATE_OR_AMOUNT column. If you do not specify an actual usage rate or amount, the resource rate or amount is derived from the WIP_OPERATION_RESOURCES table.

Similarly, you can charge person-type resources at an actual usage rate or amount. If you do not specify a value in the USAGE_RATE_OR_AMOUNT column and the STANDARD_RATE_FLAG is set to Yes (1), the resource rate or amount is derived from the WIP_OPERATION_RESOURCES table. If no usage rate or amount is specified for person type resources, and the STANDARD_RATE_FLAG is set to No (2), the system uses the employees hourly labor rate from the WIP_EMPLOYEE_LABOR_RATES table to derive the usage rate or amount. If an invalid employee ID is specific, the record is not processed.

Outside Processing Currency Option

For outside processing resource transactions (PO Move and PO Receipt charge type resources), you can specify both the currency and the resource to which your transaction is charged. You specify the currency of the outside processing resource transaction in the CURRENCY_CODE column. If this currency code is different from the base currency of the organization that you are transacting the resource in, then you must specify the currency conversion rate between the transaction currency and your organizations base currency. You can also specify the resource rate for the transaction in the CURRENCY_ACTUAL_RESOURCE_RATE column. This rate should be in the same currency entered in the CURRENCY_CODE column. If you enter a value for the CURRENCY_ACTUAL_RESOURCE_RATE, the resource is charged using this value rather than the standard rate of the resource.

Validating Resource Transactions

The Cost Manager program groups your resource transaction rows in WIP_COST_TXN_INTERFACE and launches Cost Workers to process each group. The Cost Worker program calls the WIP Transaction Validation Engine program to validate and derive data for each of the derived columns. If data have been entered in certain derived and optional columns, the WIP Transaction Validation Engine program also validates these columns.

The system considers the dependencies among the columns in the interface table and only processes columns once they are dependent upon pass validation or are successfully derived. For example, the resource validator only validates WIP_ENTITY_NAME after ORGANIZATION_ID has been validated against ORGANIZATION_CODE.

The system creates rows in the WIP_TXN_INTERFACE_ERRORS table for each failed validation. Each row in the WIP_TXN_INTERFACE_ERRORS table contains the TRANSACTION_ID of the failed resource transaction, the name of the column that failed validation, and a brief error message stating the cause of the validation failure. Because of the dependencies between columns, the resource validator does not try to validate a column if the column that it depends on fails validation. Thus, WIP_ENTITY_NAME is not validated if ORGANIZATION_CODE is invalid.

Columns that are independent of each other can be validated regardless of the status of the other columns. For example, WIP_ENTITY_NAME is validated even if REASON_NAME is invalid because WIP_ENTITY_NAME is not dependent on REASON_NAME. Thus, the system can create multiple error records for each resource

or overhead transaction. You can then resolve multiple problems at the same time, thereby increasing the speed and efficiency of your error resolution process.

Resolving Failed Rows

Viewing Failed Rows

You can view both pending and failed resource and overhead transaction rows in the WIP_COST_TXN_INTERFACE table using the Pending Resource Transactions window. You also can view the errors associated with failed transactions by navigating to the Pending Resource Transaction Errors window.

Fixing Failed Rows

You use the Pending Resource Transactions window to update failed resource and overhead transaction rows in the WIP_COST_TXN_INTERFACE table. After you make any necessary changes, you enter a check mark in the Resubmit check box and save your work. The transaction row is then eligible to be picked up by the Cost Manager for revalidation and processing. If you choose Select All for Resubmit from the Tools Menu, the system checks the Resubmit check box for all queried rows. When you save your work, all of these rows become eligible for revalidation and processing.

You also use the Pending Resource Transactions window to delete problem rows from the WIP_COST_TXN_INTERFACE table. Deleting these rows ensures that you do not have duplicate data when you reload the corrected data from the source application.

You can query again transactions that fail initial validation, then resubmit them after you correct the cause of the failure. For example, if the resource transaction fails because the status of the job is Unreleased, you can use the Discrete Jobs window to release the job, then resubmit the pending resource transaction. See: Processing Pending Resource Transactions, *Oracle Work in Process User's Guide.*

Work Order Interface

The Work Order Interface enables you to import Discrete job and Repetitive schedule header information, and Discrete job operations, material, resource, and scheduling information from any source, using a single process.

You can import:

- Planned orders for new Discrete jobs,
- Discrete job operations, components, resources, resource usage, and scheduling details
- Update and reschedule recommendations for existing Discrete jobs
- Suggested Repetitive schedules

Work in Process then uses this information to automatically create new Discrete jobs

and pending Repetitive Schedules, or to update existing Discrete jobs.

The Work Order Interface consists of two tables: the WIP_JOB_SCHEDULE_INTERFACE table (Open Job and Schedule Interface table), and the WIP_JOB_DTLS_INTERFACE table (WIP Job Details Interface table). You load header information into the WIP_JOB_SCHEDULE_INTERFACE table, and operations, components, resources, and scheduling information into the WIP_JOB_DTLS_INTERFACE table.

> **Note:** The WIP_SCHEDULING_INTERFACE table is now merged with the WIP_JOB_DTLS_INTERFACE table and thus no longer exists.

Major features

- **Record insertion from any source**: you can insert records into the Work Order Interface from any source including bar code readers, automated test equipment, cell controllers, and other manufacturing execution systems, planning systems, order entry systems, finite scheduling packages, production line sequencing programs, spreadsheets, and even custom entry forms. If, for example, your plant directly feeds to your customers plant, you can take demands directly from your customer rather than waiting for the next MRP run, thus reducing response time and eliminating unnecessary overhead.

- **Automatic Discrete job creation from Oracle Advanced Planning and Scheduling**: if you have installed Oracle Advanced Planning and Scheduling (APS), you can use its High Level Scheduling Engine to schedule Work in Process job resources for planned work orders, then import the work orders into Work in Process to create new Discrete jobs or Repetitive schedules, or to update existing Discrete jobs. APS passes the job, its bill of material, routing, components, resources, and resource usage, start and end times to the appropriate Work Order Interface table.

- **Explosion option for bills of material and routings**: you can turn on or off the bill of material (BOM) and the routing explosion feature when you load Discrete jobs or Repetitive schedules. If you turn the option on, the system uses the standard system-generated BOM and routing; if you turn the option off, you must provide a custom BOM and routing and enter them manually.

- **Job schedule at creation time**: you can schedule a Discrete job or Repetitive schedule at the time that you create it. If you choose not to schedule it at that time, then the schedule dates are imported from the Job Details Interface table.

- **Single process import of sets of material or resource requirements**: you can change specific sets of operations, components, and resources in a single process.

- **Import of operation, material, resource, and resource usage details**: you can add or change operations, components, operation resources, and update (replaces

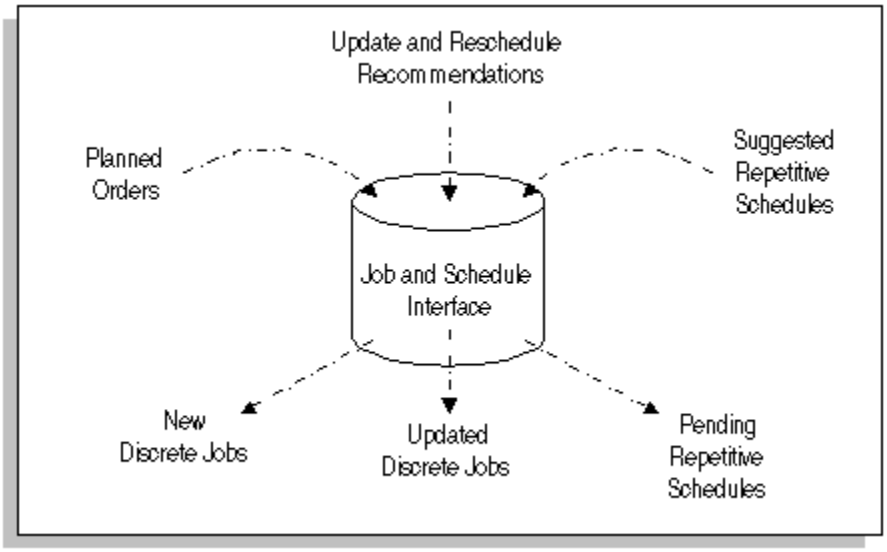
existing resource usage with the resource usage in the table) operation resource usage on Discrete jobs, as indicated on the following table.

Task Performed	New Discrete Job	Existing Discrete Job
Add Header	Currently supported	Not applicable
Add Operations	New Feature	New Feature
Change Operations	New Feature	New Feature
Delete Operations	Not supported	Not supported
Add Components	New Feature	Currently supported
Change Components	New Feature	Currently supported
Delete Components	Not supported if explosion flag is No	Currently supported
Add Operation Resources	New Feature	Currently supported
Change Operation Resources	New Feature	Currently supported
Delete Operation Resources	Not supported if explosion flag is No	Currently supported
Update* Operation Resource Usage	New Feature	New Feature

* Replaces existing operation resource usage with the resource usage entered in the interface table.

Functional Overview

The graphic depicts the types of inputs that the Work Order Interface supports as well as their corresponding Work in Process Discrete job or Repetitive schedule output.



You insert records from third party sources into the Work Order Interface tables by:

- Writing a PL/SQL program or SQL script that maps your source files to the columns in the Work Order Interface tables
- Using a third party program that can map your source files to the interface tables
- Using Oracle Advanced Planning and Scheduling's High Level Scheduling Engine to schedule planned and unreleased work orders and import them into Work in Process.
- Using Planner Workbench to automatically import planned work orders into Work in Process (see: Overview of the Planner Workbench, Implementing Planning Recommendations, and Netting Supply and Demand.

When you insert records into the Work Order Interface, you load header information into the WIP_JOB_SCHEDULE_INTERFACE table, and operations, components, resources, resource usage, and scheduling information into the WIP_JOB_DTLS_INTERFACE table. You then use the Import Jobs and Schedules window to launch the WIP Mass Load (WICMLX) concurrent program, which validates records in the Work Order Interface table and imports them into Work in Process.

The records are processed in batches identified by a group identifier (GROUP_ID) that you assign them when you launch the WIP Mass Load program. The imported records are then automatically implemented as new or updated Discrete jobs, or pending Repetitive schedules. Jobs are automatically scheduled, unless you set the flag for the scheduling method (SCHEDULING_METHOD) to manual or the flag that determines whether the bill of material and routing are exploded (ALLOW_EXPLOSION) to No before importing the records.

The WIP Mass Load program does all of the following:

- Validates the data in the interface tables
- Derives values for additional columns
- Creates new Discrete jobs, updates existing jobs, and creates pending Repetitive schedules
- Optionally launches the Job and Schedule Interface Report (WIPMLINT), which lists both successfully processed and failed records.
- Deletes successfully processed records from the interface table

Related Topics

Work Order Interface Report, *Oracle Work in Process User's Guide*

Importing Jobs and Schedules, *Oracle Work in Process User's Guide*

Processing Pending Jobs and Schedules, *Oracle Work in Process User's Guide*

Setting Up the Work Order Interface

The Work Order Interface requires no additional setup steps beyond those already required to set up Discrete and Repetitive manufacturing. All processing is initiated through the Import Jobs and Schedules window. The concurrent Import Jobs and Schedules Request that you submit through this window is managed by the Standard Manager, which thus must be set up and running.

Related Topics

Discrete Manufacturing Parameter, *Oracle Work in Process User's Guide*

Repetitive Manufacturing Parameters,, *Oracle Work in Process User's Guide*

Inserting Records Into the Work Order Interface

In order to insert records (rows) from your source system into the Work Order Interface tables, you must write a PL/SQL program, a SQL script, or use a third party program. The following sections provide you with the information that you need to map your source files to the columns in the Work Order Interface tables:

Creating New Work Orders

When you insert records into the Work Order Interface to create new or update existing Discrete jobs, you have the option to insert header information only, detail information only, or both header and detail information. You also can insert header information to create pending Repetitive schedules.

You insert header information into the WIP_JOB_SCHEDULE_INTERFACE table, and operation, material, resource, and scheduling information into the WIP_JOB_DTLS_INTERFACE table. Once you load the job header, the WIP Mass Load

program loads detail records from the WIP_JOB_DTLS_INTERFACE table that match the header record.

You can create a new work order with or without exploding the bill of material (BOM) and routing. If you choose NOT to explode the BOM and routing, you must manually enter the jobs components, operations, resources, and scheduling information in the WIP_JOB_DTLS_INTERFACE table.

> **Note:** If you want to change the scheduled start or completion date without the explosion feature, then you must provide both the FIRST_UNIT_START_DATE and the LAST_UNIT_COMPLETION_DATE.) If you choose to explode the BOM and routing, Work in Process uses the standard system BOM and routing.

Inserting header information:

When you insert header information into the WIP_JOB_SCHEDULE_INTERFACE table, you set the appropriate LOAD_TYPE to either 1 for standard Discrete jobs, 2 for Repetitive schedules, or 4 for non-standard Discrete jobs.

Inserting job or schedule details:

When you insert operation, component, resource, and scheduling details into the WIP_JOB_DTLS_INTERFACE table, you set the LOAD_TYPE to either 1 (load/update resources), 2 (load/update components), 3 (load/update operations), or 4 (load resource usage). The WIP_JOB_DTLS_INTERFACE table must contain a job header row if you are loading job schedule details.

Standard Versus Non-Standard Discrete Jobs

The WIP Mass Load program calculates start and completion dates in non-standard discrete jobs differently than discrete jobs. For non-standard discrete jobs, with operations and resources:

- If the Allow Explosion flag is set to Yes, either the start or completion date is used if the other date is null.
- If the Allow Explosion flag set to No, and values are provided for both start and completion dates, the job is rescheduled from completion date (that is, it is backward scheduled).
- If the Allow Explosion flag set to No, and either start or completion date is provided, the WIP Mass load program fails because the other date cannot be derived.

Updating Existing Work Orders

You can insert header information only, detail information only, or both header and detail information into the Work Order Interface tables, as follows:

- **Updating header information only:** To update work orders, you load the data into the WIP_JOB_SCHEDULE_INTERFACE table, and set the LOAD_TYPE of the table to 3 (update Discrete job). The work order is identified by its name if it is a Discrete job, and by its assembly or start date if it is a Repetitive schedule.

- **Updating detail information only:** To update detail records, you load the data into the WIP_JOB_DTLS_INTERFACE table. The PARENT_HEADER_ID must be null, and you must provide the WIP_ENTITY_ID and ORGANIZATION_ID. You can add, change, or delete components and operation resources on Discrete jobs. Deleting a resource also deletes all of its resource usage information. You also can update Discrete job resource usage sets, which deletes the existing resource usage and replaces it with the resource usage in the table.

  Note: You can insert an operation if a null header row is added to the wip_job_schedule_interface.

- **Updating both header and detail information:** load the data into both the WIP_JOB_SCHEDULE INTERFACE and WIP_JOB_DTLS_INTERFACE tables.

WIP_JOB_SCHEDULE_INTERFACE Table

The following table lists the columns in the WIP_JOB_SCHEDULE_ INTERFACE table and provides their load/update type and validation information

Number	Load/Update Type Description
1	Create Standard Discrete Job
2	Create Pending Repetitive Schedule
3	Update Standard or Non-Standard Discrete Job
4	Create Non-Standard Discrete Job

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
ALLOW_EXPLOSION	Varchar2 (1)	-	1,3,4	-	-	If set to N, must provide requirements manually; any values other than N or n are assumed to be Y.
ALTERNATE_BOM_DESIGNATOR	Varchar2 (10)	-	-	1,4	2,3	Copied to WIP_DISCRETE_JOBS on creation. Ignored for Repetitive schedules and during reschedule. Used by scheduler and exploder. If Alternate Routing is not defined for this assembly or reference; it will error if Assembly Type or Routing Type is 2, and profile option WIP: See Enginerring Items is set to No.
ALTERNATE_ROUTING_DESIGNATOR	Varchar2 (10)	-	-	1,4	2,3	See ALTERNATE_BOM_DESIGNATOR.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
ATTRIBUTE1 - ATTRIBUTE15	Varchar2 (150)	-	-	1,2,3,4	-	Copied to WIP_DISCRETE_JOBS and WIP_REPETITIVE_SCHEDULES on creation. Updates any NOT NULL segments from interface table.
ATTRIBUTE_CATEGORY	Varchar2 (30)	-	-	1,2,3,4	-	Descriptive flexfield structure defining column. See ATTRIBUTE1 - ATTRIBUTE15.
BOM_REFERENCE_ID	Number	-	-	4	1,2,3	If LOAD_TYPE = 1, values ignored and warning issued. Must exist in MTL_SYSTEM_ITEMS for your organization, and must have MTL_SYSTEM_ITEMS flags set correctly for WIP.
BOM_REVISION	Number	-	1,2,4	-	3	If NULL, derived from REVISION_DATE. If NOT NULL, revision must be valid for Assembly or Reference. If both REVISION and REVISION_DATE entered, must match. Ignored on reschedule; issues warning.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
BOM_REVISION_DATE	Date	-	1,2,4	-	3	Dates must correspond to valid revisions if entered. If NULL and REVISION NULL, defaulted to greater: start_date, or SYSDATE. If NULL and REVISION NOT NULL, defaults to high revision date for REVISION. If both REVISION and REVISION_DATE entered, must match. Ignored on reschedule; issues warning.
BUILD_SEQUENCE	Number	-	-	1,3,4	2	Schedule group ID and build sequence combination must be unique across all jobs. Cannot have build sequence unless have schedule group ID. Ignored for Repetitive. If NOT NULL, issues warning. Defers errors on records that fail validation.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
CLASS_CODE	Varchar2 (10)	4	1	-	2,3	If NULL on standard job creation, uses default class from WIP parameters. Ignored on reschedule and derived for Repetitive schedules. If NOT NULL, issues warning. Defers errors on records that fail validation.
COMPLETION_LOCATOR_ID	Number	-	1,4	-	2,3	Default completion locator for Discrete job.
COMPLETION_LOCATOR_SEGMENTS	Varchar2 (10)	-	1,4	-	2,3	-
COMPLETION_SUBINVENTORY	Varchar2 (10)	-	1,4	-	2,3	Default Completion Subinventory for the Discrete job.
CREATED_BY	Number	1,2,3,4	-	-	-	Standard Who column. See Required Columns.
CREATED_BY_NAME	Varchar2 (100)	1,2,3,4	-	-	-	Standard Who column. See Required Columns.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
CREATION_DATE	Date	1,2,3,4	-	-	-	Has a NOT NULL restriction; not loaded into the table; default is SYSDATE.
DAILY_PRODUCTION_RATE	Number	2	-	-	1,3,4	Ignored for Discrete jobs; warning given if NOT NULL.
DEMAND_CLASS	Varchar2 (30)	-	-	1,2,4	3	Copied to WIP_DISCRETE_JOBS and WIP_REPETITIVE_SCHEDULES on creation. Ignored on job reschedule.
DESCRIPTION	Varchar2 (240)	-	1,2,4	3	-	Copied to WIP_DISCRETE_JOBS, WIP_REPETITIVE_SCHEDULES, and WIP_ENTITIES on creation. If NULL, defaulted to generic description including date mass loaded. Derived for Repetitive schedules. Warning given if NOT NULL.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
DUE_DATE	Date	-	-	1,3,4	-	Date job to be completed (could be different from the scheduled completion date); also date used when rescheduling jobs. Default is scheduled completion date. Copied to WIP_DISCRETE_JOBS on creation and update. Ignored by Repetitive schedules.
DUE_DATE_PENALTY	Number	-	-	1,3,4	-	The penalty (in days) incurred if job completes later than the requested completion date.
DUE_DATE_TOLERANCE	Number	-	-	1,3,4	-	The number of days late a job can be schedule from the requested due date.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
FIRM_PLANNED_FLAG	Number	-	1,2,4	3	-	Copied to WIP_DISCRETE_JOBS or WIP_REPETITIVE_SCHEDULES on creation. If column NULL, value defaulted. Value must be 1, 2 (Y or N); error occurs if value is 1 and creating nonstandard job. Must be NOT NULL on reschedule. Errors on records that fail validation are deferred.
FIRST_UNIT_COMPLETION_DATE	Date	1,2,4	-	3	-	See FIRST_UNIT_START_DATE.
FIRST_UNIT_START_DATE	Date	1,2,4	-	3	-	If SCHEDULING_METHOD is manual, first unit start date (FUSD) must be less than or equal to last unit completion date (LUCD). If SCHEDULING_METHOD is routing-based, then you must enter FUSD or LUCD. Dates entered must exist in BOM_CALENDAR_DATES.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
GROUP_ID	Number	1,2,3,4	-	-	-	Identifies a batch of records for processing; when loading detail records, this column cannot be NULL (see Control Columns).
HEADER_ID	Number	1,2,3,4	-	-	-	Identifies individual jobs in a given group and ties a header record to a set of detail records. Cannot be NULL when loading detail records. Ignored by Repetitive schedules.
INTERFACE_ID	Number	-	-	-	1,2,3,4	Number generated by Work in Process to uniquely identify each record in the table; also links interface records with error records. Must be NULL (values are entered when record picked up by WIP Mass Load program).

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
JOB_NAME	Varchar2 (240)	-	1,4	-	2,3	Copied to WIP_DISCRETE_JOBS on creation. If NULL on job creation, defaulted using prefix and sequence. Either ID or Name must be entered on reschedule; if both, must match. Job Name must be unique within organization. Ignored by Repetitive schedules.
KANBAN_CARD_ID	Number	-	-	-	1,2,3,4	Only used if Oracle Inventory inserts replenishment kanban signals into the table; otherwise ignored, thus do not include.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
LAST_UNIT_COMPLETION_DATE	Date	1,2,4	-	3	-	If SCHEDULING_METHOD is manual, first unit start date (FUSD) must be less than or equal to last unit completion date (LUCD). If SCHEDULING_METHOD is routing-based, then you must enter FUSD or LUCD. Dates entered must exist in BOM_CALENDAR_DATES.
LAST_UNIT_START_DATE	Date	1,2,4	-	3	-	See LAST_UNIT_COMPLETION_DATE.
LAST_UPDATE_DATE	Date	1,2,3,4	-	-	-	Has a NOT NULL restriction; is not loaded into interface table; SYSDATE is used.
LAST_UPDATE_LOGIN	Number	-	-	1,2,3,4	-	Standard Who column. Load the value for this column in WIP_DISCRETE_JOBS.
LAST_UPDATED_BY	Number	1,2,3,4	-	-	-	Standard Who column. See Required Columns.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
LAST_UPDATED_BY_NAME	Varchar2 (100)	1,2,3,4	-	-	-	Standard Who column. See Required Columns.
LINE_CODE	Varchar2 (10)	2	-	1,3,4	-	If both LINE_ID and LINE_CODE entered, name is ignored and warning given. If only name entered, ID is derived from WIP_LINES. ID must exist and be active in WIP_LINES in correct organization. If entered or derived ID is NULL, error occurs. Defers errors on records that fail validation.
LINE_ID	Number	2	-	1,3,4	-	See LINE_CODE.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
LOAD_TYPE	Number	1,2,3,4	-	-	-	Indicates whether current interface record is planned order, job update recommendation, or suggested Repetitive schedule. Must assign one of these values or error occurs. 1 Create standard Discrete job 2 Create Pending Repetitive schedule 3 Update standard or non-standard Discrete job 4 Create non-standard Discrete Job See Control Columns for more information.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
LOT_NUMBER	Varchar2 (80)	-	1,4	3	2	If assembly under lot control, copied to WIP_DISCRETE_JOBS on creation. If NULL on job creation and parameter = Based on Job, defaults from job name; if parameter = Inventory, defaults from Inventory. Interface value ignored for Repetitive schedules during reschedule and when assembly not under lot control.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
NET_QUANTITY	Number	-	-	1,3,4	2	Copied to WIP_DISCRETE_JOBS on creation or reschedule. Rounded to six places. Must be greater than or equal to zero; must be less than or equal to START_QUANTITY. Must be zero for nonstandard jobs without an assembly. If NULL on job reschedule, assumes quantity unchanged. Ignored for Repetitive schedules; warning given if NOT NULL. Defers errors on records that fail validation.
ORGANIZATION_CODE	-	-	-	-	-	See Required Columns. When rescheduling Discrete jobs, ORGANIZATION_CODE, ORGANIZATION_ID, and WIP_ENTITY_ID, WIP_ENTITY_NAME must match record in WIP_DISCRETE_JOBS table or error message issued.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
ORGANIZATION_ID	-	-	-	-	-	Identifier for the organization. See ORGANIZATION_CODE and Required Columns.
OVERCOMPLETION_TOLERANCE_TYPE	Number	-	-	1,2,3,4	-	-
OVERCOMPLETION_TOLERANCE_VALUE	Number	-	-	1,2,3,4	-	-
PRIMARY_ITEM_ID	Number	1, 2	-	4	3	Required for standard jobs and repetitive schedules. Ignored on reschedule (warning issued). BUILD_IN_WIP_FLAG must be Y, PICK_COMPONENTS _FLAG and ENG_ITEM_FLAG must be N.
PRIORITY	Number	-	-	1,3,4	-	Processing order of the work order being imported; cannot be less than zero. Copied to WIP_DISCRETE_JOBS on creation and update. Ignored by Repetitive schedules.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
PROCESS_PHASE	Number	1,2,3,4	-	-	-	See Control Columns.
PROCESS_STATUS	Number	1,2,3,4	-	-	-	See Control Columns.
PROCESSING_WORK_DAYS	Number	2	-	-	1,3,4	Ignored for Discrete jobs; warning given if NOT NULL.
PROGRAM_APPLICATION_ID	Number	-	-	-	1,2,3,4	Extended Who column. See Derived or Ignored Columns.
PROGRAM_ID	Number	-	-	-	1,2,3,4	Extended Who column. See Derived or Ignored Columns.
PROGRAM_UPDATE_DATE	Date	-	-	-	1,2,3,4	Extended Who column.
PROJECT_ID	Number	-	3	1, 4	2	Project reference for the Discrete job.
PROJECT_NUMBER	Varchar2 (25)	-	3	1, 4	2	Project reference for the Discrete job.
REQUEST_ID	Number	-	-	-	1,2,3,4	Extended Who column.See Derived or Ignored Columns.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
REPETITIVE_SCHEDULE_ID	Number	-	-	-	1,2,3,4	Identifier for a Repetitive schedule. Must be unique across all organizations. Use sequence to generate if NULL. Warning issued if NOT NULL. Ignored for Discrete jobs.
ROUTING_REFERENCE_ID	Number	-	-	4	1,2,3	If LOAD_TYPE = 1, values ignored and warning issued. Must exist in MTL_SYSTEM_ITEMS for your organization, and must have MTL_SYSTEM_ITEMS flag set correctly for WIP.
ROUTING_REVISION	Number	-	1,2,4	-	3	Derived from REVISION_DATE if NULL. If NOT NULL, revision must be valid for Assembly or Reference. If both REVISION and REVISION_DATE entered, must match. Ignored on reschedule; warning issued.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
ROUTING_REVISION_DATE	Date	-	1,2,4	-	3	Dates must correspond to valid revisions if entered. If NULL and REVISION NULL, defaulted to greater: start_date, or SYSDATE. If NULL and REVISION NOT NULL, default to high revision date for REVISION. If both REVISION and REVISION_DATE entered, must match. Ignored on reschedule; warning issued.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
SCHEDULE_GROUP_ID	Number	-	-	1,3,4	2	If both SCHEDULE_GROUP_NAME and SCHEDULE_GROUP_ID are entered, name ignored and warning issued. If only name entered, ID derived from WIP_SCHEDULE_GROUPS. If entered or derived ID is NULL, error occurs. ID must exist and be active in WIP_SCHEDULE_GROUPS in correct organization. Ignored for Repetitive schedules. If NOT NULL, warning issued. Defers errors on records that fail validation.
SCHEDULE_GROUP_NAME	Varchar2 (240)	-	-	1,3,4	2	See SCHEDULE_GROUP_ID.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
SCHEDULING_METHOD	Number	-	1,3,4	-	2	Valid values are: Routing-Based, Item Lead Time, Manual. If NULL, default is Routing-Based; if Repetitive, must be Routing-Based; if Routing-Based, all operation and resource dates set to start date.
SOURCE_CODE	Varchar2 (30)	-	-	1,2,3,4	-	Values copied into WIP_DISCRETE_JOBS during Discrete mass load. If NOT NULL, enter the value for this column when rescheduling Discrete jobs and Repetitive schedules.
SOURCE_LINE_ID	Number	-	-	1,2,3,4	-	See SOURCE_CODE.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
START_QUANTITY	Number	-	-	1,3,4	2	Copied to WIP_DISCRETE_JOBS on creation or reschedule; rounded to six places. Error issued if NULL on job creation. Must be greater than zero when creating standard Discrete jobs or greater than or equal to zero when creating nonstandard jobs. If NULL when rescheduling, assumes quantity unchanged. Ignored for Repetitive schedules. Warns if NOT NULL. Defers errors on records that fail validation.
STATUS_TYPE	Number	-	1,4	3	2	Must be one of the following: 1 Unreleased 3 Released 6 On Hold
TASK_ID	Number	-	3	1, 4	2	See Optional/Derived if Null Columns.
TASK_NUMBER	Varchar2 (25)	-	3	1, 4	2	See Optional/Derived if Null Columns.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
WIP_ENTITY_ID	Number	3	-	-	1,2,4	WIP Discrete job or Repetitive assembly identifier. Copied to WIP_DISCRETE_JOBS on job creation. Must be unique across all organizations. Ignored by Repetitive schedules. Either ID or Name must be entered on reschedule. If both entered, must match. If NULL on creation, generated using a sequence. When rescheduling Discrete jobs, WIP_ENTITY_ID, WIP_ENTITY_NAME, and ORGANIZATION_ID, ORGANIZATION_CODE must match record in WIP_DISCRETE_JOBS table or you receive an error message.
WIP_ENTITY_NAME	VarChar2 (100)	3	-	-	1,2,4	See WIP_ENTITY_ID.

Column	Type	Required	Optional/Derived if Null	Optional	Derived or Ignored	Additional Information
WIP_SUPPLY_TYPE	Number	-	1, 4	-	2,3	Method of material consumption within WIP. Used to explode BOM on job creation; copied to WIP_DISCRETE_JOBS. Must be valid supply type (1-5, 7; cannot choose 2 or 3 for non-standard jobs without an assembly). If NULL, value defaults to Based on Bill. Ignored for Repetitive schedules and during reschedule.
SERIALIZATION_START_OP	Number	-	-	-	-	This should be populated if the user wants to use serialized transactions.

Subinventory and Locator Columns

To null Supply Subinventory and Completion Subinventory values for push type components using WIP Mass Load, populate subinventory values in the WIP_JOB_DTLS_INTERFACE table with fnd_api.g_miss_char.

To null Supply Locator and Completion Locator values, populate locator values with fnd_api.g_miss_num.

Control Columns

The following columns are control columns for the WIP Mass Load program. Other columns in the interface represent the actual data that are inserted into or modified in the WIP_DISCRETE_JOBS and WIP_REPETITIVE_ SCHEDULES tables when records are successfully imported from the Work Order Interface table. You can find more information about the WIP_DISCRETE_JOBS and WIP_REPETITIVE_ SCHEDULES

tables in the *Oracle Work in Process eTechnical Reference Manual*.

- ALLOW_EXPLOSION: determines whether the system uses the standard bill of material (BOM) and routing or a custom BOM and routing that you supply. If this flag is set to N or n, you must manually provide a custom BOM and routing; otherwise the system uses the standard BOM and routing.

- GROUP_ID: identifies the batch of detail records loaded. It is used to group records (rows) in the interface table for processing. Since Work in Process only processes records with a GROUP_ID, you must assign the records a GROUP_ID when you launch the WIP Mass Load program. You use the WIP_JOB_SCHEDULE_INTERFACE_S sequence to generate a new, unique GROUP_ID for each batch of rows that you insert into the WIP_JOB_SCHEDULE_INTERFACE table. Work in Process does NOT process records that have a NULL GROUP_ID. If GROUP_ID is NULL, the record stays in the interface table as a reference.

- HEADER_ID: identifies individual jobs in a given group, and ties a header record to a set of detail records.

- INTERFACE_ID: identifies each work order that is loaded

- LOAD_TYPE: determines whether the current interface record is a planned order, update recommendation, or suggested Repetitive schedule. It also controls whether interface table columns are Required, Optional, Optional/Derived if Null, or Derived or Ignored, and has a NOT NULL restriction. You must assign one of the following possible values or an error occurs:
  - 1 Create Standard Discrete Job
  - 2 Create Pending Repetitive Schedule
  - 3 Update Standard or Non-Standard Discrete Job
  - 4 Create Non-Standard Discrete Job

- PROCESS_PHASE: together with PROCESS_STATUS, the PROCESS_PHASE column indicates the current status of each record. Possible PROCESS_PHASE values include:
  - 2 Validation
  - 3 Explosion
  - 4 Completion
  - 5 Creation

- PROCESS_STATUS: together with PROCESS_PHASE, the PROCESS_STATUS column indicates the current status of each record. Possible PROCESS_STATUS values include:
  - 1 Pending
  - 2 Running
  - 3 Error
  - 4 Complete
  - 5 Warning

    Records should be inserted into the WIP_JOB_SCHEDULE_INTERFACE table with a PROCESS_PHASE = 2(Validation) and a PROCESS_STATUS = 1 (Pending). These values indicate that the record is ready to be processed by the WIP Mass Load program. If the program fails at any stage when processing a record, the PROCESS_STATUS of that record is set to 3 (Error). Records that load successfully have their PROCESS_STATUS set to 4 (Complete). If a record fails to load because, for example, the WIP Mass Load program is abnormally terminated, the PROCESS_STATUS of the record is set to 5 (Warning). To resubmit these records, you set the PROCESS_STATUS of status 5 (Warning) records to 1 (Pending), and set the PROCESS_PHASE to 2 (Validation), then resubmit them.

Required Columns

You must specify values for columns in this category. If you do not enter a required value, the WIP Mass Load program does not process the record and inserts an error record in the WIP_INTERFACE_ERRORS table. If you specify values for both the name and the ID, the value for the ID is used and the value for the name is ignored during validation. If the entered or derived ID is NULL, you receive an error. Errors on records that fail validation are deferred.

- CREATED _BY, CREATED_BY_NAME, and LAST_UPDATED_BY, LAST_UPDATED_BY_NAME:If you enter only the name, the ID is derived from FND_USER and therefore must exist and be active in FND_USER.

- LINE_CODE, LINE_ID: If you only enter the code, the ID is derived from WIP_LINES. The ID must exist and be active in WIP_LINES in the correct organization.

- ORGANIZATION_CODE, ORGANIZATION_ID: If you only enter the code, the ID is derived from ORG_ORGANIZATION_DEFINITIONS, thus, the ID must exist and be active in ORG_ORGANIZATION_DEFINITIONS.

  Both Work in Process and Oracle Inventory parameters must be defined for the organization. When rescheduling Discrete jobs, the ORGANIZATION_CODE,

ORGANIZATION_ID, WIP_ENTITY_ID, and WIP_ENTITY_NAME must match the record in the WIP_DISCRETE_JOBS table or you will receive an error message.

Optional or Derived if Null Columns

You have the option to specify values for columns that the WIP Mass Load program will use. If you leave Optional/Derived if Null columns blank (NULL), the program uses an internal default value instead. For example, for records with a LOAD_TYPE of 1 (Create Discrete Jobs), the STATUS_TYPE field is Optional/Derived if Null. If you specify a value, that value is used when the job is created. If you do not specify a value, the value defaults to 1 (Unreleased). In general, default values are derived in the same way that they are derived when you manually enter Discrete jobs and Repetitive schedules in the Discrete Jobs and Repetitive Schedules windows. (See: Defining Discrete Jobs Manually and Defining Repetitive Schedules Manually, *Oracle Work in Process User's Guide*.)

For some optional columns (SCHEDULE_GROUP_ID, SCHEDULE_GROUP_NAME, PROJECT_ID, PROJECT_NUMBER, and TASK_ID, TASK_NUMBER), you can use either the name or the underlying ID. If you specify values for both the name and the ID, the value for the ID is used and the value for the name is ignored.

- PROJECT_ID, PROJECT_NUMBER and TASK_ID, TASK_NUMBER: these columns are interdependent. When loading records that update existing jobs (Load/Update Type #3), the following rules are applied.

Task = Null and Project = Null

If the job has a project and task reference, then the values in these fields are not overwritten, and thus remain unchanged

Project < > Null and Task = Null

If the job has a project and task reference, the project number is overwritten and the task is overwritten with the NULL task

> **Note:** If the Project Level Reference parameter in the Organization Parameters window in Oracle Inventory is set to task and a task is required, then records with only a project will fail to load

Project = Null and Task < > Null

If the job has a project, the project field is not overwritten with the NULL project. If the job has a task, however, the task is overwritten.

> **Note:** Records with new tasks will only successfully load if those tasks have been associated with the jobs project in Oracle Projects.

- Completion locators for standard project jobs (Load/Update Type #3) are automatically recreated when a project or task changes.

- STATUS_TYPE: you can only create Discrete jobs with a status of 1 Unreleased, 3

Released, or 6 On Hold. If the column is NULL when a job is created, Unreleased is the default. Repetitive schedules are created with a status of 13 Pending Mass Loaded.

> **Note:** If you have installed Oracle Manufacturing Scheduling, and are using the Constraint-Based Scheduling engine, Unreleased is the only option available.

Only valid status changes can occur when you are rescheduling jobs. If the column is NULL when a job is rescheduled, there is no status change. A job must have a released status or be rescheduled to a released state to be released. If it is rescheduled to an unreleased status from a released status, it is not released.

Optional Columns

You do not have to enter values for columns in this category. Unlike Optional/Derived if Null columns, however, Optional columns are not defaulted if left blank. The same validation logic that is applied when you manually enter values for these fields in the Discrete Jobs and Repetitive Schedules windows is applied to the values that you enter in these columns.

For some optional columns (SCHEDULE_GROUP_ID, SCHEDULE_GROUP_NAME, PROJECT_ID, PROJECT_NUMBER, and TASK_ID, TASK_NUMBER), you can use either the name or the underlying ID. If you specify values for both the name and the ID, the value for the ID is used and the value for the name is ignored.

Derived or Ignored Columns

These columns are for internal processing only. You should leave all columns in this category blank (NULL), since values entered in these columns are ignored or overwritten.

- PROGRAM_APPLICATION_ID, PROGRAM_ID, REQUEST_ID: unlike other Derived or Ignored columns, the Mass Load Program does not set any values for these columns, nor does it copy them when it creates or reschedules Discrete jobs or Repetitive schedules. Thus, you must enter the values for these columns in the interface table both when creating and rescheduling Discrete jobs and Repetitive schedules.

WIP_JOB_DTLS_INTERFACE Table

When you load operations and detailed component, resource, and scheduling information for Discrete jobs from your source files into Work in Process, it loads the data in batches, based on their GROUP_ID. The tables PARENT_HEADER_ID column connects the work order details back to the header information in the Open Job and Schedule Interface. You must set the LOAD_TYPE to one of the following:

1. Load a resource
2. Load a component

3. Load an operation
4. Load multiple resource usage
5. Change between primary and alternate resources
6. Load an operational link
7. Associate/Disassociate serial numbers
8. Dispatch resource instance
9. Load a resource instance usage

You can add or change Discrete job operations, components, and resources. You also can delete operation resources and components on existing Discrete jobs, however, you can only delete them on *new* Discrete jobs if the ALLOW_EXPLOSION Flag is set to Yes (indicates that you are using the standard system BOM and routing). If you delete operation resources, you need to provide the value for OPERATION_SEQ_NUM and RESOURCE_SEQ_NUM. If you delete components, you must provide the COMPONENT_SEQ_ID and the WIP_ENTITY_ID (you must provide the OPERATION_SEQ_ID only if you attach the component to an operation).

The following table lists the columns in the WIP_JOB_DTLS_INTERFACE table, and indicates whether they are Required (Reqd), Optional (Opt), or Null when you add or change operations, components, and resources for existing Discrete jobs, or when you schedule new Discrete jobs and pending Repetitive schedules. When adding operations, components, or resources, use Substitution Type 2; when changing them, use Substitution Type 3. Do not provide columns marked Null.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
ACTIVITY_ID	Number	Null	Null	Null	Null	Opt	Opt	Null	Identifier for the activity
APPLIED_RESOURCE_UNITS	Number	Null	Null	Null	Null	Null	Null	Null	Number of resource units charged.
APPLIED_RESOURCE_VALUE	Number	Null	Null	Null	Null	Null	Null	Null	Value of the resource units charged.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
ASSIGNED_UNITS	Number	Null	Null	Null	Null	Reqd	Opt	Null	Number of resource units assigned to do work.
ATTRIBUTE1 through ATTRIBUTE15	VarChar2 (150)	Opt	Opt	Opt	Opt	Opt	Opt	Null	Descriptive flexfield segments
ATTRIBUTE_CATEGORY	VarChar2 (30)	Opt	Opt	Opt	Opt	Opt	Opt	Null	Descriptive flexfield structure defining column.
AUTOCHARGE_TYPE	Number	Null	Null	Null	Null	Reqd	Opt	Null	Method of charging the resource.
BACKFLUSH_FLAG	VarChar 2 (1)	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations; optional when updating.
BASIS_TYPE	Number	Null	Null	Null	Null	Reqd	Opt	Null	Basis for scheduling and charging the resource.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
COMPLETION_DATE	Date	Null	Null	Null	Null	Reqd	Opt	Null	Resources scheduled finished production date. Required for loading and updating resource usage. If USAGE_BLOCK is specified, then completion date pertains to it. To change scheduled completion date without explosion, must also provide first unit start date and last unit completion date.
COMPONENT_YIELD_FACTOR	Number	Null	Null	Opt	Opt	Null	Null	Null	-
COUNT_POINT_TYPE	Number	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations; optional when updating.
CREATED_BY	Number	-	-	-	-	-	-	-	Standard Who column.
CREATION_DATE	Date	-	-	-	-	-	-	-	Standard Who column.
DATE_REQUIRED	Date	Null	Null	Reqd	Opt	Null	Null	Null	-

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
DEPARTMENT_ID	Number	Reqd	Opt	Opt	Opt	Null	Null	Null	Department identifier.
DESCRIPTION	VarChar2 (240)	Opt	Opt	Opt	Opt	Opt	Opt	Opt	-
FIRST_UNIT_COMPLETION_DATE	Date	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations; optional when updating.
FIRST_UNIT_START_DATE	Date	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations; optional for updating. To change schedule start or completion date without explosion, must also provide first unit start date and last unit completion date.
GROUP_ID	Number	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	If details pertain to header record loaded at same time, value must be same as GROUP_ID in WIP_JOB_SCHEDULE_INTERFACE table.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
INTERFACE_ID	Number	Null	Null	Null	Null	Null	Null	Null	Number generated by Work in Process to uniquely identify each record in the table. Links interface records with error records. Must be NULL (values are entered when record picked up by WIP Mass Load program).
INVENTORY_ITEM_ID_NEW	Number	Null	Null	Reqd	Opt	Null	Null	Null	-
INVENTORY_ITEM_ID_OLD	Number	Null	Null	Null	Reqd	Null	Null	Null	-
ITEM_SEGMENTS	VarChar (2000)	Opt	Opt	Opt	Opt	Opt	Opt	Opt	-
LAST_UNIT_COMPLETION_DATE	Date	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations; optional for updating. To change schedule start or completion date without explosion, must also provide first unit start date and last unit completion date.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
LAST_UNIT_START_DATE	Date	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations; optional when updating.
LAST_UPDATE_DATE	Date	Reqd	-	-	-	-	-	-	Standard Who column.
LAST_UPDATE_LOGIN	Number	Reqd	-	-	-	-	-	-	Standard Who column.
LAST_UPDATED_BY	Number	Reqd	-	-	-	-	-	-	Standard Who column.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
LOAD_TYPE	Number	3	3	2	2	1	1	4	Cannot be NULL. 1 for loading a resource 2 for loading a component 3 for loading an operation 4 for loading multiple resource usage 5 for changing between primary and alternate resources 7 for associating serial numbers 8 for dispatching resource instance 9 for loading resource instance usage
MINIMUM_TRANSFER_QUANTITY	Number	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations optional when updating.
MPS_DATE_REQUIRED	Date	Null	Null	Opt	Opt	Null	Null	Null	Date used by MPS relief process.
MPS_REQUIRED_QUANTITY	Number	Null	Null	Opt	Opt	Null	Null	Null	Quantity used by MPS relief process.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
MRP_NET_FLAG	Number	Null	Null	Reqd	Opt	Null	Null	Null	Determines whether or not MRP should consider the component requirement in its netting process.
OPERATION_SEQ_NUM	Number	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Number of operation sequence within a routing. When adding a component, the operation sequence defaults to the first operation of the job if this value is set to 1.
ORGANIZATION_ID	Number	Null	Null	Null	Null	Null	Null	Null	Identifier for the organization.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
PARENT_HEADER_ID	Number	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Contains HEADER_ID of work order record (GROUP_ID, HEADER_ID columns from WIP_JOB_SCHEDULE_INTERFACE table identify header record uniquely). Must be NULL if only detail records are loaded or updated. Must provide WIP_ENTITY_ID and ORGANIZATION_ID.
PROCESS_PHASE	Number	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	See Control Columns.
PROCESS_STATUS	Number	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	Reqd	See Control Columns.
PROGRAM_APPLICATION_ID	Number	-	-	-	-	-	-	-	Extended Who column.
PROGRAM_ID	Number	-	-	-	-	-	-	-	Extended Who column.
PROGRAM_UPDATE_DATE	Date	-	-	-	-	-	-	-	Extended Who column.
QUANTITY_ISSUED	Number	Null	Null	Reqd	Opt	Null	Null	Null	Part quantity issued.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
QUANTITY_PER_ASSEMBLY	Number	Null	Null	Reqd	Opt	Null	Null	Null	Part usage quantity.
REQUEST_ID	Number	-	-	-	-	-	-	-	Extended Who column.
REQUIRED_QUANTITY	Number	Null	Null	Reqd	Opt	Null	Null	Null	Part quantity required.
RESOURCE_ID_NEW	Number	Null	Null	Null	Null	Reqd	Opt	Null	-
RESOURCE_ID_OLD	Number	Null	Null	Null	Null	Null	Reqd	Null	-
RESOURCE_SEQ_NUM	Number	Null	Null	Null	Null	Reqd	Reqd	Reqd	Number of the resource sequence.
SCHEDULED_FLAG	Number	Null	Null	Null	Null	Reqd	Opt	Null	Method of scheduling the resource.
STANDARD_RATE_FLAG	Number	Null	Null	Null	Null	Reqd	Opt	Null	Determines whether or not resource is charged at standard rate.
STANDARD_OPERATION_ID	Number	Opt	Opt	Null	Null	Null	Null	Null	Optional.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
START_DATE	Date	Null	Null	Null	Null	Reqd	Opt	Null	Required for loading and updating resource usage. If USAGE_BLOCK is specified, then start date pertains to it. To change schedule start date without explosion, must also provide first unit start date and last unit completion date.
SUBSTITUTION_TYPE	Number	-	-	-	-	-	-	-	Must be one of these: 1 Delete 2 Add 3 Change Any other values will cause an error.
SUPPLY_LOCATOR_ID	Number	Null	Null	Opt	Opt	Null	Null	Null	Locator used to supply component to WIP.
SUPPLY_SUBINVENTORY	VarChar2(10)	Null	Null	Opt	Opt	Null	Null	Null	Subinventory used to supply component to WIP
USAGE_BLOCK	-	Null	Null	Null	Null	Null	Null	Reqd	Required for loading and updating resource usage.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
USAGE_RATE_OR_AMOUNT	Number	Null	Null	Null	Null	Reqd	Opt	Null	Rate per assembly or amount per Discrete job or Repetitive schedule.
UOM_CODE	VarChar2(3)	Null	Null	Null	Null	Reqd	Opt	Null	Code for the unit of measure.
WIP_ENTITY_ID	Number	Null	Null	Null	Null	Null	Null	Null	Value generated when job loaded, thus must be NULL in this table.
WIP_SUPPLY_TYPE	Number	Null	Null	Reqd	Opt	Null	Null	Null	Method of material consumption within WIP.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
SERIAL_NUMBER_NEW	VarChar2(30)	Null	Null	Null	Null	Null	Null	Null	This column is used to associate serial numbers to a job. During the job creation, the user can check the Auto Associate Serial Numbers On Job Creation From Interface in the WIP parameters form to automatically generate and associate serials to the job (enough serials for the job quantity). If you are using this column, use load type 7.

Column Name	Type	Op Add	OpChg	Cmp Add	Cmp Chg	Res Add	Res Chg	Sched	Additional Information
SERIAL_NUMBER_OLD	VarChar2(30)	Null	Null	Null	Null	Null	Null	Null	This column is used to de-associate serial numbers to a job. During job creation, the user can check the Auto Associate Serial Numbers On Job Creation From Interface in the WIP parameters form to automatically generate and associate serials to the job (enough serials for the job quantity). If you are using this column, use load type 7.

Control Columns

- GROUP_ID: used to group (batch) rows in the interface table. Only records with a GROUP_ID are processed by LOAD_WIP.

- PROCESS_PHASE: must be 2 (validation) for Work in Process to pick up the record and process it. (Records loaded into the table by LOAD_INTERFACE are assigned a process phase of 1. Only records with a process phase of 2 are picked up by LOAD_WIP. Therefore, either your third party scheduling program or your custom program must change the process phase to a 2.)

- PROCESS_STATUS: must be 1 (pending) for Work in Process to pick up the record and process it. It will be updated to 2 (running) when the record is being processed, and to 6 (complete) when the record is loaded to Work in Process successfully, or 3 (error) if it fails.

- SUBSTITUTION_TYPE: must be either 1 (delete) or 2 (add) or 3 (change). It will error out for all other values.

Primary and Alternate Resources

The Work Order Interface provides the ability to change between primary and alternate resources. To enable this feature:

- Insert a header record into Wip_Job_Schedule_Interface table for the job
- Next, insert a child record in Wip_Job_Dtls_Interface with the following values:
  - load_type = 1 (resource)
  - substitution_type = 3 (update)
  - replacement_group_num—set this value to a valid replacement group number of the primary resource's substitute group
- For the following columns, set the values for the primary resource in the Wip_Operation_Resources table):
  - operation_seq_num
  - resource_seq_num
  - resource_id_old
  - resource_id_new
  - substitute_group_num

Validating Work Order Interface Records

The WIP Mass Load program validates all required and optional data. If the required or optional data that you enter are invalid, or if required data are missing, the program updates the PROCESS_STATUS for the record to 3 (Error), and an error message tied to the row's interface ID is inserted into the WIP_INTERFACE_ERRORS table. Unsuccessfully processed rows that have a PROCESS_STATUS of 3 (Error) can be viewed, updated, deleted, or resubmitted using the Pending Jobs and Schedules window.

Data in the WIP_JOB_DTLS_INTERFACE table that must be added, deleted, or changed are first validated against WIP constraints. If any records in the WIP_JOB_DTLS_INTERFACE table fail, all records for that Discrete job fail. All records for a failed Discrete job are checked to ensure that there is a specific error message for each failed row. Other SQL fatal errors are passed to the calling program.

Viewing Failed Rows

You can view information on both pending and failed rows in the Pending Jobs and Schedules window and view errors associated with the failed rows by navigating to the

Pending Job and Schedule Errors window.

You also can obtain information on failed rows by printing the *Work Order Interface Report*. If you print the Work Order Interface Report as part of the import process, both successfully and unsuccessfully processed rows are listed. Successfully processed rows are deleted from the WIP_JOB_SCHEDULE_INTERFACE table after the Work Order Interface Status Report is submitted for printing.

Resolving Failed Rows

Use the Pending Jobs and Schedules window to update failed rows in the WIP_JOB_SCHEDULE_INTERFACE table. After you make the changes, enter a check mark in the Resubmit check box and save your work. If you Select All for Resubmit from the *Tools Menu*, the system checks the Resubmit check box for all queried rows. After saving, all of these rows become eligible for revalidation and processing.

You use the Pending Jobs and Schedules window to delete problem rows from the WIP_JOB_SCHEDULE_INTERFACE table. Deleting these rows ensures you do not have duplicate data when you reload the corrected data from the source.

Related Topics

Oracle Work in Process eTechnical Reference Manual

Processing Pending Jobs and Schedules, *Oracle Work in Process User's Guide*

13

Oracle Inventory Open Interfaces and APIs

This chapter covers the following topics:

- Open Transaction Interface
- Transaction Flow Application Program Interface
- Open Replenishment Interface
- Cycle Count Entries Interface
- Cycle Count Application Program Interface
- Kanban Application Program Interface
- Lot Application Program Interface
- Material Reservation Application Program Interface
- Reservations Manager Application Program Interface
- Sales Order Application Program Interface
- Move Order Application Program Interface
- Move Order Admin API
- Line_Details_pub
- Pick Confirm Application Program Interface
- Quantity Tree Program Interface
- Material Transactions Applications Program Interface
- Inter-Company Transaction Flow Application Program Interface
- Serial Number Public Application Program Interface
- User Defined Serial Generation Application Program Interface

Open Transaction Interface

Oracle Inventory provides an open interface for you to load transactions from external applications and feeder systems. These transactions could include sales order shipment transactions from an Order Management system other than Oracle Order Management, or they could be simple material issues, receipts, or transfers loaded from data collection devices. The following transaction types are supported by this interface:

- Inventory issues and receipts (including user-defined transaction types)
- Subinventory transfers
- Direct interorganization transfers
- Intransit shipments
- WIP component issues and returns
- WIP assembly completions and returns
- Sales order shipments
- Inventory average cost updates
- LPN Pack
- Unpack
- Split Transactions
- Inventory Lot Split/ Merge/ Translate Transactions

This interface is also used as an integration point with Oracle Order Management for shipment transactions. Oracle Order Management's Inventory Interface program populates the interface tables with transactions submitted through the Confirm Shipments window.

You must write the load program that inserts a single row for each transaction into the MTL_TRANSACTIONS_INTERFACE table. For material movement of items that are under lot or serial control, you must also insert rows into MTL_TRANSACTION_LOTS_INTERFACE and MTL_SERIAL_NUMBERS_INTERFACE respectively. If you insert WIP assembly/completion transactions that complete or return job assemblies, you must also insert rows into the CST_COMP_SNAP_ INTERFACE table if the organization referenced uses average costing. The system uses this information to calculate completion cost.

There are two modes you can use to process your transactions through the interface. In

the first processing mode, you populate the interface table only. Then the Transaction Manager polls the interface table asynchronously looking for transactions to process, groups the transaction rows, and launches a Transaction Worker to process each group. In the second processing mode, you insert the rows in the interface table and call a Transaction Worker directly, passing the group identifier of the interfaced transactions as a parameter so that the worker can recognize which subset of transactions to process.

The Transaction Worker calls the Transaction Validator, which validates the row, updates the error code and explanation if a validation or processing error occurs, and derives or defaults any additional columns.

Next, the Transaction Processor records the transaction details in the transaction history table along with relevant current cost information. All material movement transactions update inventory perpetual balances for the issue, receipt, or transfer locations.

Once the transaction has been successfully processed, the corresponding row is deleted from the interface table. Finally, the transaction is costed by the transaction cost processor, which runs periodically, picking up all transactions from the history table that have not yet been marked as costed.

Additional Transaction Processing Flow Steps

The following transactions require additional processing by the transaction processor or other modules.

- Inventory Issue Transactions: Inventory Issue transactions consume any existing reservations where the Transaction Source Type and Source match. For example, if you reserved 10 boxes of paper for the Finance department, and then you issue 4 boxes to that department, the reservation will automatically be partially consumed, with a remaining balance of 6 reserved boxes.

- Average Cost Transactions: In average cost organizations, receipts and average cost update transactions modify the item's average cost using the current average cost, on hand quantity, and the transaction value and quantity (if appropriate) to calculate the new average.

- WIP Issue Transactions: WIP issue transactions also update quantity issued for all material requirements on the job or repetitive schedule and charge the costs of issued components to the job/schedule.

- WIP Completion Transactions: WIP completion transactions update the job or repetitive schedule completed quantities, launch appropriate backflush transactions, and relieve costs of completed assembly from the job/schedule. If you are completing an ATO assembly, you must specify the sales order demand details so that Oracle Inventory can reserve the completed units to the appropriate sales order line/shipment.

  If the WIP completion/return transaction completes or return job assemblies in an average costing organization, the rows in the CST_COMP_SNAP_INTERFACE

table are transferred to the CST_COMP_SNAPSHOT table so that completion costs can be calculated.

- Sales Order Shipment Transactions: For sales order shipment transactions, the Transaction Processor attempts to consume any reservations that may have been created for an order by matching the Order, Line, Delivery, and Picking Line identifiers. If MRP is installed, the processor also creates an interface row in MRP_RELIEF_INTERFACE that the MRP Planning Manager uses to relieve the Master Demand Schedule.

- LPN Based Transactions: The following LPN-based transactions are supported for material that is in Inventory (not supported for material in 'Receiving' or that is 'staged'):

  - Receive into LPN: This is with a standard inventory-receipt transaction by populating the LPN_ID column in the interface table. The LPNs used for this transaction must be pre-generated by the WMS Container API.

  - Issue of an LPN: This is with the standard inventory-issue transaction by populating the CONTENT_LPN_ID column in the interface table.

  - Pack into an LPN: This is done by inserting a transaction of action Pack for the item to be packed and by populating the TRANSFER_LPN_ID column. The LPNs used for this transaction must be pre-generated by the WMS Container API.

  - Movement of an LPN between subinventory/locations: This is done by inserting a transaction of type Subinventory Transfer and populating CONTENT_LPN_ID with the LPN that is being moved.

  - Unpack from an LPN: This is done by inserting a transaction of action Unpack for the item to be unpacked and by populating the LPN_ID column.

  - Split part of contents of one LPN into another LPN in the same location: This is done by inserting a transaction of action Split for the item to be transferred between LPNs. LPN_ID is to be populated with the LPN from where the material is to be removed. TRANSFER_LPN_ID columns must be populated with the LPN into which material is to be packed.

- Inventory Lot Split/Merge/Translate Transactions: The following Inventory Lot Split/Merge/Translate transactions are supported through Open Interface for Lot controlled items and Lot and Serial controlled items:

  - Inventory Lot Split: This is done by splitting a single lot for item under lot and serial control into two or more new lots with serial numbers of the parent lot split across the child lots. The lot attributes defined on the original lot are copied onto the newly created lots. The serial numbers remain the same and

hence the serial attributes are available for items in the new lots. Serial number information cannot be changed or updated while performing the transaction.

- Inventory Lot Merge: This is done by merging two or more lots into a completely new lot or an already existing lot with the serial numbers of the merging lots combined on to the merged lot. Two lots containing different serial numbers under lot and serial control are merged into a single lot retaining the same serial number. Serial number information cannot be changed or updated while performing the transaction.

- Inventory Lot Translate: This is done by the following ways:
  - Same lot for different items. Item is under lot and serial control.
  - Different lot for the same item.
  - Different lot for a different item that is also under lot and serial control.

  In all of the above three methods, the serial numbers of the parent lot are retained and no new serial numbers can be generated while translating the lot.

Lot and Serial Transaction Detail Relationships

If you are transacting items under lot and/or serial control, you need to link the lot/serial transaction detail rows to their parent row. You accomplish this by populating MTL_TRANSACTIONS_INTERFACE. TRANSACTION_INTERFACE_ID with a unique value to be used as the primary key to link the child lot/serial rows. If the item is under lot control, you populate the foreign key MTL_TRANSACTION_ LOTS_INTERFACE.TRANSACTION_INTERFACE_ID with the same value for all child lot rows of the transaction and ensure that the total of all the lot quantities adds up to the transaction quantity on the parent row. Similarly, if the item is under serial control, you populate the foreign key MTL_SERIAL_NUMBERS_INTERFACE. TRANSACTION_INTERFACE_ID with the value in the parent row and ensure that the total number of serial numbers adds up to the transaction quantity of the parent row.

If the item is under both lot and serial control, the serial interface rows must belong to lot parent rows. This means that the relationship between MTL_TRANSACTIONS_INTERFACE and MTL_ TRANSACTION_LOT_NUMBERS remains the same as in the case where the item is under only lot control, but you also need to populate each lot row with a unique value in MTL_TRANSACTION_LOT_ NUMBERS.SERIAL_TRANSACTION_TEMP_ID. You then need to populate the foreign key MTL_SERIAL_NUMBERS_INTERFACE. TRANSACTION_INTERFACE_ID with the value in the parent lot row and ensure that the total number of serial numbers adds up to the lot quantity in the parent row.

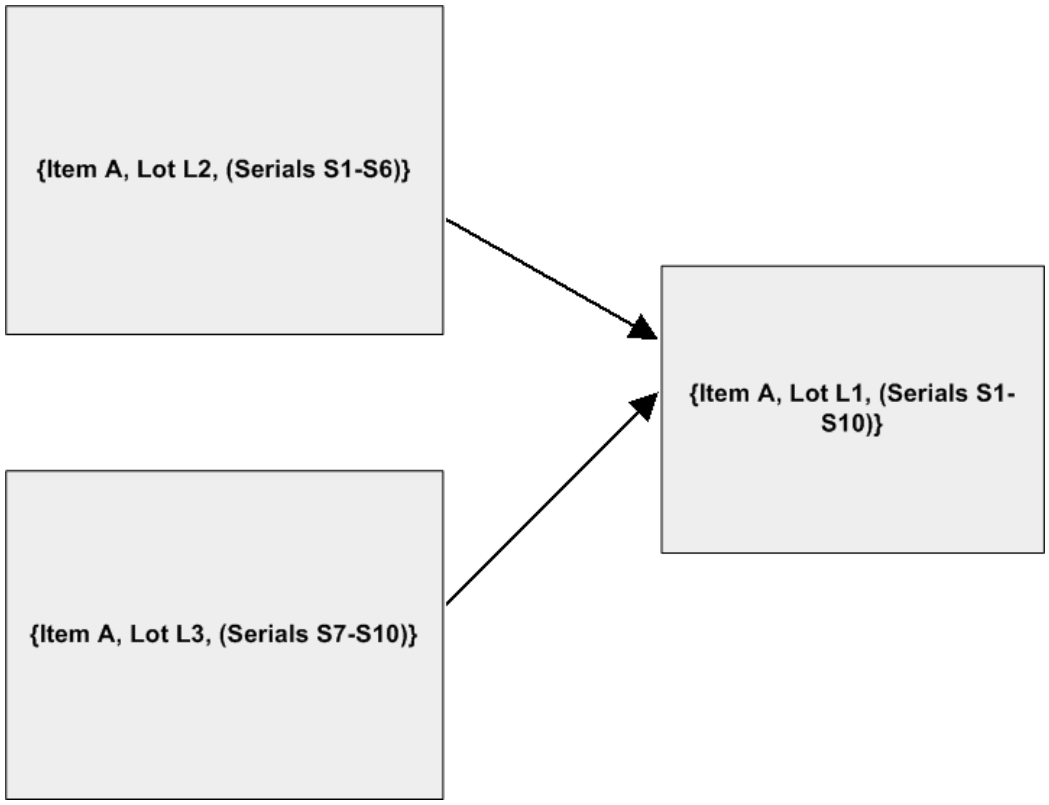
Completion Cost Detail Relationships

If you are completing or returning WIP assembly items for a job in an average costing organization, you need to link the completion cost detail rows to their parent rows. You accomplish this by populating MTL_TRANSACTIONS_INTERFACE. TRANSACTION_INTERFACE_ID with a unique value to be used as the primary key to link the child completion cost rows. You must also populate the foreign key CST_COMP_SNAP_INTERFACE.TRANSACTION_INTERFACE_ID with the same value for all child completion cost calculation rows.

Performing Lot Transactions through Open Interface

Lot Merge Transaction

Two or more lots can be merged into a completely new lot or an already existing lot with the serial numbers of the merging lots combined on to the merged lot. Two lots L1 and L2 containing serials S1-S6 and S7-S10 respectively for item A that is under lot and serial control can be merged into a single lot L1 with serials S1-S10. And serial number information cannot be changed/updated while performing the transaction. The Lot Merge transaction is diagrammatically depicted by figure 7-2 below:

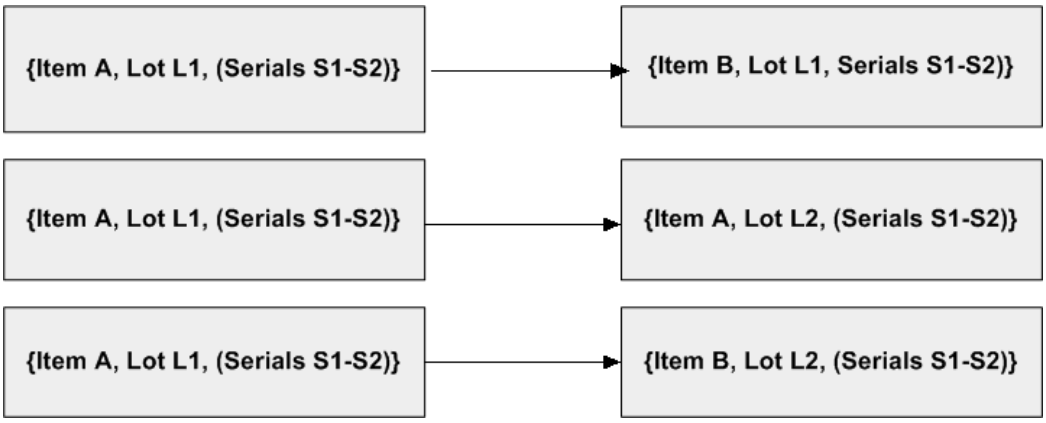


Lot Translate Transaction

A lot L1 for Item A under lot and serial control containing 2 serials S1 and S2 can be translated in the following 3 ways:

- Same lot L1 for a different item, say Item B that is also under lot and serial control.
- Different lot L2 (New lot) for the same item A.
- Different lot L2 (New lot) for a different item B that is also under lot and serial control.

In all the above three cases the serial numbers S1and S2 of the parent lot L1 are retained and no new serial numbers can be generated while translating the lot. Serial Uniqueness constraints are honored when start and resultant items are different. The Lot Translate transaction is depicted in figure 7-3 below:



Inserting into Interface tables for Lot Split/ Merge/ Translate transactions

To complete the transaction successfully, both the source and resultant records should be inserted into MTL_TRANSACTIONS_INTERFACE, MTL_TRANSACTION_LOTS_INTERFACE and MTL_SERIAL_NUMBERS_INTERFACE in a certain manner. The columns that group the records and have a specific relation with respect to the other records in the group are described below.

For example:

If Lot L1 containing serials (S1-S4) splits into two lots L2 with serials (S1-S2) and L3 with serials (S3-S4):

- Three records need to be inserted into MTL_TRANSACTIONS_ INTERFACE, one for each lot.
- Three records need to be inserted into MTL_TRANSACTION_LOTS_INTERFACE corresponding to the record inserted into MTL_TRANSACTIONS_ INTERFACE.
- Three records need to be inserted into MTL_SERIAL_NUMBERS_INTERFACE,

The records will be in the following order:

1. Specifying the fm_serial_number, to_serial_number as S1, S4 respectively for Lot L1.
2. Specifying the fm_serial_number, to_serial_number as S1, S2 respectively for Lot L2.
3. Specifying the fm_serial_number, to_serial_number as S3, S4 respectively for Lot L3.

MTL_TRANSACTIONS_INTERFACE

- PARENT_ID: This column identifies the parent record in a batch of records inserted

for the transaction. Parent_id for all the 3 records (for L1 split to L2 and L3) should be populated with the TRANSACTION_ INTERFACE_ID of the source record i.e. of Lot L1.

- TRANSACTION_BATCH_ID: The source and resultant records should belong to the same batch, as if any one of these records fail any validations, the whole batch should fail. So this column should be populated with the same value for source and resultant records in the MTL_TRANSACTIONS_INTERFACE table.

- TRANSACTION_BATCH_SEQ: This value orders the transaction records that belong to a batch .The source lot for a lot split/lot translate should have a lower transaction_batch_seq value than the resultant lot and in case of Lot Merge transaction, resultant lot should have a lower transaction_batch_seq value than the source lots. For example:

  - For Lot Split transaction (L1->L2, L3), the value of transaction_batch_seq is 1 for the source lot L1, and 2,3 for the resultant lots L2, L3 respectively.

  - For Lot Merge transaction (L2, L3->L1) the value of transaction_batch_seq is 2,3 for source lots L2, L3 respectively, and 1 for resultant lot L1.

  - For Translate transaction (L1->L2) the value of transaction_batch_seq for source lot L1 is 1, and 2 for resultant lot L2.

- TRANSACTION_QUANTITY: You must enter a negative value for the source records and positive value for the resultant records for Lot Split/Merge/Translate transactions for transaction quantity.

MTL_TRANSACTION_LOTS_INTERFACE

- LOT_NUMBER: You must insert one record per lot as specified by this column into the table, which should correspond to the record inserted into MTL_TRANSACTIONS_INTERFACE.

- TRANSACTION_QUANTITY: The transaction quantity entered for this record should match the transaction_quantity of the corresponding record in MTL_TRANSACTIONS_INTERFACE. This should be of negative value for the source record and positive value for the resultant records.

MTL_SERIAL_NUMBERS_INTERFACE

- FM_SERIAL_NUMBER/TO_SERIAL_NUMBER: You can use the from_serial_number and to_serial_number if you want to transfer a range of serial numbers.

  - If Lot L1 containing Serials S1 to S10 is split into Lots L2 (S1 to S5) and L3 (S6 to S10), then for the source Lot L1 from_serial_number=S1 and

to_serial_number=S10, for the first resultant Lot L2 from_serial_number=S1 and to_serial_number=S5, for second resultant Lot L3 from_serial_number=S6 and to_serial_number=S10

- If Lot L1 with serials (S1 to S4) and Lot L2 with Serials (S6 to S10) are merged into Lot L1 with Serials (S1 to S4, S6 to S10), for first source Lot L1 from_serial_number=S1 and to_serial_number=S4, second source Lot L2 from_serial_number=S6 and to_serial_number=S10 .For the resultant Lot L3 2 records need to be inserted with the first record having from_serial_number=S1 and to_serial_number=S4 and the second record having from_serial_number=S6 and to_serial_number=S10.

- For Lot Translate transaction the serial numbers should not be populated as serial numbers from the source lot are copied to the resultant lot.

Serial Attributes

The serial attributes are copied on to the resultant lot from source lot and they can be changed by populating the SERIAL_ATTRIBUTE_CATEGORY and the corresponding serial attributes of the resultant MTL_SERIAL_NUMBERS_INTERFACE records.

Lot Split Transaction

Lot L1 for Item A containing serials (S1 to S10) is split into Lot L2 with serials (S1 to S5) and Lot L3 with Serials (S6 to S10).

	MTI	**MTLI**	**MSNI**
Source	TRANSACTION_BATCH_SEQ = 1TRANSACTION_BATCH_ID = 100ITEM_SEGMENT1 = 'A'TRANSACTION_QUANTITY = -10	TRANSACTION_QUANTITY = -10LOT_NUMBER = 'L1'	FM_SERIAL_NUMBER = 'S1'TO_SERIAL_NUMBER = 'S10'
Resultant1	TRANSACTION_BATCH_SEQ = 2TRANSACTION_BATCH_ID = 100ITEM_SEGMENT1 = 'A' TRANSACTION_QUANTITY = 5	TRANSACTION_QUANTITY = 5LOT_NUMBER = 'L2'	FM_SERIAL_NUMBER = 'S1'TO_SERIAL_NUMBER = 'S5'

Resultant2	TRANSACTION_BATCH_SEQ = 3TRANSACTION_BATCH_ID = 100ITEM_SEGMENT1 = 'A' TRANSACTION_QUANTITY = 5	TRANSACTION_QUANTITY = 5LOT_NUMBER = 'L3'	FM_SERIAL_NUMBER = 'S6'TO_SERIAL_NUMBER = 'S10'

Lot Merge Transaction

Lot L2 with serials (S1 to S5) and L3 with Serials (S6 to S10) for Item A are merged into Lot L1 with serials (S1 to S10).

	MTI	**MTNI**	**MSNI**
Source 1	TRANSACTION_BATCH_SEQ = 2TRANSACTION_BATCH_ID = 102ITEM_SEGMENT1 = 'A'TRANSACTION_QUANTITY = -5	TRANSACTION_QUANTITY = -5LOT_NUMBER = 'L2'	FM_SERIAL_NUMBER = 'S1'TO_SERIAL_NUMBER = 'S5'
Source 2	TRANSACTION_BATCH_SEQ = 3TRANSACTION_BATCH_ID = 102ITEM_SEGMENT1 = 'A'TRANSACTION_QUANTITY = -5	TRANSACTION_QUANTITY = -5LOT_NUMBER = 'L3'	FM_SERIAL_NUMBER = 'S6'TO_SERIAL_NUMBER = 'S10'
Resultant	TRANSACTION_BATCH_SEQ = 1TRANSACTION_BATCH_ID = 102ITEM_SEGMENT1 = 'A' TRANSACTION_QUANTITY = 10	TRANSACTION_QUANTITY = 10LOT_NUMBER = 'L1'	FM_SERIAL_NUMBER = 'S1'TO_SERIAL_NUMBER = 'S10'

Lot Translate Transaction

Lot L1 for Item A with serials (S1 to S10) is translated into Lot L2 for Item A with serials (S1 to S10).

	MTI	MTLI
Source	TRANSACTION_BATCH_SEQ = 1	TRANSACTION_QUANTITY = -10
	TRANSACTION_BATCH_ID = 101	LOT_NUMBER = 'L1'
	ITEM_SEGMENT1 = 'A'	
	TRANSACTION_QUANTITY = -10	
Resultant	TRANSACTION_BATCH_SEQ = 2	TRANSACTION_QUANTITY = 10
	TRANSACTION_BATCH_ID = 101	LOT_NUMBER = 'L2'
	ITEM_SEGMENT1 = 'A'	
	TRANSACTION_QUANTITY = 10	

Setting Up the Transaction Interface

Setting Up the Inventory Concurrent Manager

For optimal processing in the Inventory Transaction Interface, you need to set up your concurrent manager to best handle your transaction volumes while balancing your performance requirements and your system load restrictions. Oracle Inventory ships the Transaction Manager to be run in Inventory's own concurrent manager named Inventory Manager. It is defaulted to run in the Standard work shift with Target Processes = 1 and Sleep Time of 60 seconds. See: Transaction Managers, *Oracle Inventory User's Guide*.

With this configuration, the Material Transaction Manager and all Transaction Workers that are spawned must share the same processing queue. If you have the available resources, you can substantially reduce the time to process your interfaced transactions by increasing the target processes and reducing the concurrent manager sleep time using the Concurrent Managers window. This will allow Transaction Workers to run in parallel with the Transaction Manager and with each other. See: Defining Managers and

their Work Shifts, *Oracle Applications System Administrator's Guide*.

Starting the Inventory Transaction Manager

Once you have set up the Inventory concurrent manager, you can launch the Inventory Transaction Manager in the Interface Managers window. This launches the Material Transaction manager and lets you specify the polling interval and the number of transactions to be processed by each worker. After polling the MTL_TRANSACTIONS_INTERFACE table for eligible rows, the Transaction Manager creates the necessary number of Transaction Workers to process the load. See: Launching Transaction Managers, *Oracle Inventory User's Guide*.

Submitting a Transaction Worker Directly as a Concurrent Process

The transaction worker can be directly called either from an Oracle Form or a c program. You can also launch a worker from the operating system using the Application Object library CONCSUB utility. You need to specify the following parameters in the given order.

HEADER_ID

This is the transaction_header_id that you want the worker to process. If no header id is passed the worker will assign itself.

TABLE

Pass 1 for the Interface table and 2 for the temp table.

SOURCE_HEADER_ID

This column will be used to select rows to process if HEADER_ID is not specified.

Source_Code

This column is used to select rows to process if header id is not specified.

Setting Up Your Sales Order Flexfield

Oracle Inventory uses a flexfield to hold the unique Sales Order name so that it does not need to join back to the feeder Order Management system. This means that you must set up Inventory's Sales Order flexfield (MKTS) using the Key Flexfield Segments window with enough segments so that the combination is unique across all orders. See: Key Flexfield Segments, *Oracle Flexfields User's Guide*.

For example, Oracle Order Management guarantees uniqueness within an installation, order type, and order number. Consequently, standard installation steps require that you set up three segments. If you can guarantee that one segment is sufficient (for example, Order Number), then that is all you need to enable in your flexfield definition.

When you enter shipment transactions into the interface, you should use the Sales Order segment values to identify the order. The Material Transaction Manager will

validate against MTL_SALES_ORDERS, and if the code combination does not already exist will create a new one. All references to the order number internal to Inventory in reports and inquiries will be based on this relationship.

Inserting into the Transaction Interface Tables

This section provides a chart for each interface table that lists all columns, followed by a section giving a brief description of a subset of columns requiring further explanation. The chart identifies each column's datatype and whether it is Required, Derived, or Optional. Many of the columns are conditionally required. Reference numbers corresponding to notes immediately following the table help identify the mandatory conditions.

Several of the attributes in the interface tables can be populated using either the user-friendly values or the internal identifiers. This is particularly true of flexfields, such as Item, Locator, and Distribution Account. In these cases, you have the option to specify either the flexfield segment representation or the internal identifier (for example, INVENTORY_ITEM_ID) for the required value.

If you populate the user-friendly values, the Transaction Validator will automatically validate them and derive the internal identifiers. If the translation is already available to the external system, it may be advantageous to use the internal identifiers to improve performance (see discussion below on validation).

MTL_TRANSACTIONS_INTERFACE Table

Column Name	Type	Required	Derived	Optional
SOURCE_CODE	Varchar2(30)	x		
SOURCE_LINE_ID	Number	x		
SOURCE_HEADER_ID	Number	x		
PROCESS_FLAG	Number(1)	x		
TRANSACTION_MODE	Number	x		
LOCK_FLAG	Number(1)			x
TRANSACTION_HEADER_ID	Number		x	

Column Name	Type	Required	Derived	Optional
ERROR_CODE	Varchar2(240)		x	
ERROR_EXPLANATION	Varchar2(240)		x	
VALIDATION_REQUIRED	Number			x
TRANSACTION_INTERFACE_ID	Number	x		
INVENTORY_ITEM_ID	Number	x		
ITEM_SEGMENT1 to ITEM_SEGMENT20	Varchar2(40)	x		
REVISION	Varchar2(3)	1		
ORGANIZATION_ID	Number	x		
SUBINVENTORY_CODE	Varchar2(10)	2		
LOCATOR_ID	Number	3		
LOC_SEGMENT1toLOC_SEGMENT20	Varchar2(40)	3		
TRANSACTION_QUANTITY	Number	x		
TRANSACTION_UOM	Varchar2(3)	x		
PRIMARY_QUANTITY	Number		x	

Column Name	Type	Required	Derived	Optional
TRANSACTION_DATE	Date	x		
ACCT_PERIOD_ID	Number		x	
TRANSACTION_SOURCE_ID	Number	x		
DSP_SEGMENT1toDSP_SEGMENT30	Varchar2(40)	x		
TRANSACTION_SOURCE_NAME	Varchar2(30)	x		
TRANSACTION_SOURCE_TYPE_ID	Number		x	
TRANSACTION_ACTION_ID	Number		x	
TRANSACTION_TYPE_ID	Number	x		
REASON_ID	Number			x
TRANSACTION_REFERENCE	Varchar2(240)			x
TRANSACTION_COST	Number	4		
DISTRIBUTION_ACCOUNT_ID	Number	5		
DST_SEGMENT1toDST_SEGMENT30	Varchar2(25)	5		

Column Name	Type	Required	Derived	Optional
CURRENCY_CODE	Varchar(30)			x
CURRENCY_CONVERSION_TYPE	Varchar(30)			x
CURRENCY_CONVERSION_RATE	Number			x
CURRENCY_CONVERSION_DATE	Date			x
USSGL_TRANSACTION_CODE	Varchar(30)			x
ENCUMBRANCE_ACCOUNT	Number			x
ENCUMBRANCE_AMOUNT	Number			x
VENDOR_LOT_NUMBER	Varchar2(30)			x
TRANSFER_SUBINVENTORY	Varchar2(10)	6		
TRANSFER_ORGANIZATION	Number	6		
TRANSFER_LOCATOR	Number	3,6		
XFER_LOC_SEGMENT1toXFER_LOC_SE	Varchar2(40)	3,6		
SHIPMENT_NUMBER	Varchar2(30)	7		

Column Name	Type	Required	Derived	Optional
TRANSPORTATION_COST	Number			x
TRANSPORTATION_ACCOUNT	Number			x
TRANSFER_COST	Number			x
FREIGHT_CODE	Varchar2(25)			x
CONTAINERS	Number			x
WAYBILL_AIRBILL	Varchar2(20)			x
EXPECTED_ARRIVAL_DATE	Date			x
NEW_AVERAGE_COST	Number	8		
VALUE_CHANGE	Number	8		
PERCENTAGE_CHANGE	Number	8		
DEMAND_ID	Number			9
PICKING_LINE_ID	Number			
DEMAND_SOURCE_HEADER_ID	Number			10
DEMAND_SOURCE_LINE	Varchar2(30)			10

Column Name	Type	Required	Derived	Optional
DEMAND_SOURCE_DELIVERY	Varchar(30)			10
WIP_ENTITY_TYPE	Number	11,12		
SCHEDULE_ID	Number		11,12	
OPERATION_SEQ_NUM	Number	11	12	
REPETITIVE_LINE_ID	Number	13		
NEGATIVE_REQ_FLAG	Number			x
TRX_SOURCE_LINE_ID	Number			9
TRX_SOURCE_DELIVERY_ID	Number			9
CUSTOMER_SHIP_ID	Number			x
SHIPPABLE_FLAG	Varchar2(1)		x	
LAST_UPDATE_DATE	Date	x		
LAST_UPDATED_BY	Number	x		
CREATION_DATE	Date	x		
CREATED_BY	Number	x		
LAST_UPDATE_LOGIN	Number			x

Column Name	Type	Required	Derived	Optional
REQUEST_ID	Number			x
PROGRAM_APPLICATION_ID	Number			x
PROGRAM_ID	Number			x
COST_GROUP_ID	Number	8		
PROGRAM_UPDATE_DATE	Date			x
ATTRIBUTE_CATEGORY	Varchar2(30)			x
ATTRIBUTE1 to ATTRIBUTE15	Varchar2(150)			x
BOM_REVISION	Varchar2(1)			15
BOM_REVISION_DATE	Date			15
ROUTING_REVISION	Varchar2(1)			15
ROUTING_REVISION_DATE	Date			15
ALTERNATE_BOM_DESIGNATOR	Varchar2(1)			14
ALTERNATE_ROUTING_DESIGNATOR	Varchar2(1)			14
ACCOUNTING_CLASS	Varchar2(1)			15

Column Name	Type	Required	Derived	Optional
DEMAND_CLASS	Varchar2(1)			14
PARENT_ID	Number	18		14
SUBSTITUTION_ID	Number			14
SUBSTITUTION_ITEM_ID	Number			14
SCHEDULE_GROUP	Number			14
BUILD_SCHEDULE	Number			14
REFERENCE_CODE	Number			14
FLOW_SCHEDULE	Varchar2(1)	16		
SCHEDULED_FLAG		17		
LPN_ID	Number			X
CONTENT_LPN_ID	Number			X
TRANSFER_LPN_ID	Number			X

1. If under revision control
2. All transaction types except average cost update
3. If under locator control
4. Inventory Issues and Receipts in an average cost organization

5. Inventory Issues/Receipts of an asset item to/from an asset subinventory and sales order shipment transactions
6. Inventory direct transfers (inter- or intra-organization)
7. Intransit shipments
8. Average cost update transactions only
9. Sales order shipment transactions
10. To reserve/unreserve ATO items to a sales order upon completion/return from a WIP job
11. WIP component issues/returns
12. WIP assembly completions/returns
13. Repetitive schedules
14. For work orderless completions
15. For work orderless completions, derived if null
16. Must be set to Y
17. Must be set to 2

SOURCE_CODE

This column is required for Sales Order transactions to identify the source Order Management system. For other transaction types, you can enter any useful value for tracking purposes. The values entered are transferred directly to the transaction history table.

SOURCE_HEADER_ID

You can use this column as an external system reference. The values entered are transferred directly to the transaction history table.

SOURCE_LINE_ID

You can use this column as an external system reference. The values entered are transferred directly to the transaction history table.

PROCESS_FLAG

This column controls whether rows in the interface table are processed. You should insert a row that you intend to be processed with a value of 1 (Yes). The valid values are:

1. Yes
2. No
3. Error

TRANSACTION_MODE

This column determines how the interfaced transactions will be processed. The valid options are:

2 - Concurrent

3 - Background

Interface transactions marked for Background processing will be picked up by the transaction manager polling process and assigned to a transaction worker. These transactions will not be processed unless the transaction manager is running.

You use Concurrent transaction mode if you want to launch a dedicated transaction worker to explicitly process a set of transactions. The Transaction Manager does not process transactions marked for concurrent processing.

LOCK_FLAG

The Transaction Manager uses this column to manage the worker assignment process. You should need to update this column only if a transaction has failed due to an exceptional failure such as the system going down in the middle of transaction worker processing. In this case, you will need to reset the LOCK_FLAG to 2 so your failed transactions can be reprocessed.

TRANSACTION_HEADER_ID

This column groups transactions for assignment to specific transaction workers. Depending on the value of TRANSACTION_MODE, this column is either required (concurrent mode) or derived by the transaction manager (background mode). This column maps to MTL_MATERIAL_TRANSACTIONS.TRANSACTION_SET_ID in the transaction history tables.

ERROR_CODE DERIVED

If a transaction error occurs, the Transaction Validator populates this column with short descriptive text indicating the type of error that has occurred.

ERROR_EXPLANATION DERIVED

If a transaction error occurs, the Transaction Validator populates this column with an explanation of the error. If an explanation is not provided, check the log file for details using the View Requests window.

VALIDATION_REQUIRED

You can use this flag to control whether the Transaction Validator skips certain validation steps for certain transaction types. The options are:

1 - Full validation

2 - Validate only columns required for derivation

If you leave this field null, Full validation is used.

TRANSACTION_INTERFACE_ID

This column is required for transactions of items under lot or serial control. The value in the column in this table is used to identify the child rows in the lot or serial interface tables MTL_TRANSACTION_LOTS_INTERFACE and MTL_SERIAL_NUMBERS_INTERFACE.

If the transacted item is under lot control, this column maps to MTL_TRANSACTION_LOTS_INTERFACE. TRANSACTION_INTERFACE_ID. If the transacted item is under serial control and not lot control, this column maps to MTL_SERIAL_NUMBERS_INTERFACE. TRANSACTION_INTERFACE_ID.

TRANSACTION_QUANTITY

Enter the transaction quantity in the transaction unit of measure. The quantity should be positive for receipts into inventory, and negative for both issues out of inventory and transfers. Enter a quantity of 0 for Average Cost Update transactions.

TRANSACTION_UOM

You can enter the TRANSACTION_QUANTITY in any unit of measure that has conversion rates defined to the item's primary unit of measure. Use this column to specify the transacted unit of measure even if it is the same as the primary unit of measure.

PRIMARY_QUANTITY

This column is the transaction quantity in the item's primary unit of measure calculated using TRANSACTION_QUANTITY and TRANSACTION_UOM.

ACCT_PERIOD_ID

This column is derived using the entered TRANSACTION_DATE to determine within which period the transaction occurred. The transaction date must be on or before the system date at time of transaction processing, and the transaction date must lie within the boundaries of an open period (in ORG_ACCT_PERIODS).

TRANSACTION_TYPE_ID

Enter the type of transaction you are executing. The transaction types and internal IDs supported by the interface are:

Transaction Type	Internal ID
Account Issue	01
Account Alias Issue	31
Miscellaneous Issue	32
Issue Components to WIP	35
Return Assemblies to WIP	17
Account Receipt	40
Account Alias Receipt	41
Miscellaneous Receipt	42
Return Components from WIP	43
WIP Assembly Completion	44
Subinventory Transfer	02
Direct Inter-Organization Transfer	03
Intransit Shipment	21
Average Cost Update	80
Sales Order Shipment	33
Inventory Lot Split	82
Inventory Lot Merge	83
Inventory Lot Translate	84

You can identify the TRANSACTION_TYPE_ID for user-defined transactions by selecting from MTL_TRANSACTION_TYPES where TRANSACTION_TYPE_NAME is the transaction type you wish to use.

TRANSACTION_SOURCE_TYPE_ID

This column is derived from MTL_TRANSACTION_TYPES using the value you enter in TRANSACTION_TYPE_ID.

TRANSACTION_SOURCE_NAME

This column is required for user-defined transaction source types. Enter the value of the source name, such as an order number, to be displayed on all transaction reports and inquiries.

TRANSACTION_SOURCE_ID

TRANSACTION_SOURCE_ID or the corresponding flexfield segment columns (DSP_SEGMENT1 to DSP_SEGMENT30) are required for all transaction source types other than those that are user-defined. You should enter the foreign key ID that points to the context table identified by the transaction source type.

Source Type	Foreign Key Reference
Account	GL_CODE_COMBINATIONS.CODE_COMBINATION_ID
Account Alias	MTL_GENERIC_DISPOSITIONS.DISPOSITION_ID
Job or Schedule	WIP_ENTITIES.WIP_ENTITY_ID
Sales Order	MTL_SALES_ORDERS.SALES_ORDER_ID

DSP_SEGMENT1 TO DSP_SEGMENT30

You can use these flexfield segment columns instead of TRANSACTION_SOURCE_ID to enter the more user-friendly information. For example, if the interfaced transaction is for an Issue to Account transaction type, you would enter the GL Code Combination segment values in these columns instead of putting the Code GL Code Combination ID in TRANSACTION_SOURCE_ID.

TRANSACTION_ACTION_ID

This column is derived from MTL_TRANSACTION_TYPES using the value you enter in TRANSACTION_TYPE_ID.

OPERATION_SEQ_NUM

For assembly completions and returns, this value is derived. For WIP component issues and returns with routings, this value is required. For WIP routings, enter 1.

WIP_ENTITY_TYPE

For WIP component issues and returns, and WIP assembly completions and returns, enter one of the following values:

1. Standard discrete jobs
2. Repetitive schedules
3. Non-standard discrete jobs
4. Work Order-less Schedule

REASON_ID

Use this column to specify a transaction reason from the predefined list of reasons in MTL_TRANSACTION_REASONS.

TRANSACTION_REFERENCE

Use this column to enter any transaction reference information to be displayed in transaction inquiries and reports.

TRANSACTION_COST

You can use this column to specify a transaction unit cost for average cost Inventory issues and receipts. If you leave it blank, the current system unit cost is used.

DISTRIBUTION_ACCOUNT_ID

Use this column (or the flexfield segment columns) to specify the account to charge for the cost of the Inventory transaction. It is required for user-defined transactions, and derived by the Transaction Worker based on the transaction source type and source for Account Issue/Receipt and Account Alias Issue/Receipt transactions.

DST_SEGMENT1 TO DST_SEGMENT30

You can use these flexfield segment columns instead of DISTRIBUTION_ACCOUNT_ID to enter the more user-friendly information. For example, if the interfaced transaction is for an Issue to Account transaction type, you would enter the GL Code Combination segment values in these columns instead of putting the Code GL Code Combination ID in DISTRIBUTION_ACCOUNT_ID.

CURRENCY_CODE

If your transaction cost is in a different currency than the functional currency of your set of books, enter the currency code.

CURRENCY_CONVERSION_TYPE

If you enter a currency code other than the functional currency for your set of books,

enter the conversion type.

CURRENCY_CONVERSION_RATE

If you enter a currency code other than the functional currency for your set of books, enter the conversion rate

CURRENCY_CONVERSION_DATE

Enter the currency conversion date for which the conversion rate is valid for the transaction.

VENDOR_LOT_NUMBER

Use this column as transaction reference information and/or to cross-reference supplier lot numbers against internal lot numbers.

TRANSFER_ORGANIZATION

This column is required for all inter-organization transfers. Enter the destination organization's internal ID.

TRANSFER_SUBINVENTORY

This column is required for subinventory transfers within the same organization and direct transfers from one organization to another. For these scenarios, enter the destination subinventory.

TRANSFER_LOCATOR

This column is required for subinventory transfers within the same organization and direct transfers from one organization to another when the item being transferred is under locator control in the destination subinventory. For these scenarios, enter the destination locator internal ID.

XFER_LOC_SEGMENT1-XFER_LOC_SEGMENT20

When a transfer locator is required, you can optionally use these columns instead of TRANSFER_LOCATOR when you want to use the user-friendly flexfield representation of the transfer locator instead of the internal ID.

SHIPMENT_NUMBER

This column is required for intransit shipments. It groups shipment lines in RCV_SHIPMENT_LINES under a parent shipment number in RCV_SHIPMENT_HEADERS.

The Transaction Worker will not process intransit transactions if a shipment header already exists in RCV_SHIPMENT_HEADERS that matches SHIPMENT_NUMBER. If you want to group shipment lines under the same header, you must ensure they are processed by the same worker. You can accomplish this using the concurrent processing mode, using the TRANSACTION_HEADER_ID to group your interface transactions,

and directly calling a Transaction Worker to process that group.

NEW_AVERAGE_COST

Average cost update transactions require that either NEW_AVERAGE_COST, VALUE_CHANGE, or PERCENTAGE_CHANGE be populated, depending on the type of cost update being performed.

VALUE_CHANGE

See NEW_AVERAGE_COST.

PERCENTAGE_CHANGE

See NEW_AVERAGE_COST.

DEMAND_ID

Use this column for sales order shipment transactions to identify the exact reservation row to be relieved in MTL_DEMAND. If you do not have the DEMAND_ID information, leave this column blank, and the Transaction Processor will try to match reservations to relieve by checking MTL_DEMAND to see if there are any reservations where there is a match on:

MTL_TRANSACTIONS_INTERFACE	MTL_DEMAND
ORGANIZATION_ID	ORGANIZATION_ID
INVENTORY_ITEM_ID	INVENTORY_ITEM_ID
TRANSACTION_SOURCE_TYPE_ID	DEMAND_SOURCE_TYPE_ID
TRANSACTION_SOURCE_ID	DEMAND_SOURCE_HEADER_ID
TRX_SOURCE_LINE_ID	DEMAND_SOURCE_LINE
TRANSACTION_SOURCE_NAME	DEMAND_SOURCE_NAME
TRX_DELIVERY_ID	DEMAND_SOURCE_DELIVERY

TRX_SOURCE_LINE_ID

Use this column to specify details of reservations to be relieved with an issue transaction. See DEMAND_ID.

TRX_SOURCE_DELIVERY_ID

Use this column to specify details of reservations to be relieved with an issue

transaction. See DEMAND_ID.

DEMAND_SOURCE_HEADER_ID

Use this column for completion (and returns) of ATO items from a Final Assembly Order if the quantity you are completing is to be reserved to an existing sales order. Enter values in DEMAND_SOURCE_HEADER_ID, DEMAND_SOURCE_LINE_ID, and DEMAND_SOURCE_DELIVERY_ID that match the appropriate demand rows in MTL_DEMAND. The transaction processor will automatically create a reservation for the completed quantity to that sales order.

DEMAND_SOURCE_LINE

See DEMAND_SOURCE_HEADER_ID.

DEMAND_SOURCE_DELIVERY

See DEMAND_SOURCE_HEADER_ID.

BOM_REVISION

The bill revision and date determine which version of the bill is used to explode work order-less component requirements.

ROUTING_REVISION

The routing revision and date determines which version of the routing is used to create work order-less component requirements.

ALTERNATE_BOM_DESIGNATOR

An alternate bill of material is optional if alternates have been defined for the assembly you are building.

ALTERNATE_ROUTING_DESIGNATOR

An alternate routing is optional if alternates have been defined for the assembly you are building.

PARENT_ID

This column identifies the work order-less completion interface ID.

SUBSTITUTION_ID

Use this column to specify the substitution type

3 - *Add:* Add a component at the operation.

2 - *Delete:* Delete a component from the operation.

1 - *Change:* Substitute one component for another at the operation.

4 - *Lot/Serial:* Specify lot/serial number information for items.

SUBSTITUTION_ITEM_ID

This column identifies the inventory item number of the substitute item.

SCHEDULE_GROUP

This column can specify any active schedule group.

BUILD_SEQUENCE

For future use.

REFERENCE_CODE

For future use.

Transaction Lots Interface

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
TRANSACTION_INTERFACE_ID	Number	3			**PO, ASN, REQ, INV, RMA**	**MTL_MATERIAL_TRANSACTIONS_S**
SOURCE_CODE	Varchar2(30)			x		
SOURCE_LINE_ID	Number			x		
LOT_NUMBER	Varchar2(80)	x			**PO, ASN, REQ, INV, RMA**	
LOT_EXPIRATION_DATE	Date	1				
TRANSACTION_QUANTITY	Number	x			**PO, ASN, REQ, INV, RMA**	

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
PRIMARY_QUANTITY	Number	x	x		**PO, ASN, REQ, INV, RMA**	
SERIAL_TRANSACTION_TEMP_ID	Number	2			**PO, ASN, REQ, INV, RMA**	**MTL_MATERIAL_TRANSACTIONS_S**
ERROR_CODE	Varchar2(240)		x			
LAST_UPDATE_DATE	Date	x			**PO, ASN, REQ, INV, RMA**	**SYSDATE**
LAST_UPDATED_BY	Number	x			**PO, ASN, REQ, INV, RMA**	**FND_GLOBAL.USER_ID**
CREATION_DATE	Date	x			**PO, ASN, REQ, INV, RMA**	**SYSDATE**
CREATED_BY	Number	x			**PO, ASN, REQ, INV, RMA**	**FND_GLOBAL.USER_ID**
LAST_UPDATE_LOGIN	Number	x		x	**PO, ASN, REQ, INV, RMA**	**FND_GLOBAL.LOGIN_ID**
REQUEST_ID	Number			x		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
PROGRAM_APPLICATION_ID	Number			x		
PROGRAM_ID	Number			x		
PROGRAM_UPDATE_DATE	Date			x		
PRODUCT_CODE	**Varchar2(5)**	**x**			**PO, ASN, REQ, INV, RMA**	**'RCV'**
PRODUCT_TRANSACTION_ID	**Number**	**x**			**PO, ASN, REQ, INV, RMA**	**RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID**
PROCESS_FLAG	**Varchar2(1)**			**x**		
DESCRIPTION	**Varchar2(256)**			**x**		
VENDOR_ID	**Number**			**x**		
SUPPLIER_LOT_NUMBER	**Varchar2(150)**			**x**		
TERRITORY_CODE	**Varchar2(30**			**x**		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
ORIGINATION_DATE	Date			x		
DATE_CODE	Varchar2(150)			x		
GRADE_CODE	Varchar2(150)			x		
CHANGE_DATE	Date			x		
MATURITY_DATE	Date			x		
STATUS_ID	Number			x		
RETEST_DATE	Date			x		
AGE	Number			x		
ITEM_SIZE	Number			x		
COLOR	Varchar2(150)			x		
VOLUME	Number			x		
VOLUME_UOM	Varchar2(3)			x		
PLACE_OF_ORIGIN	Varchar2(150)			x		
BEST_BY_DATE	Date			x		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
LENGTH	**Number**			**x**		
LENGTH_UOM	**Varchar2(3)**			**x**		
RECYCLED_CONTENT	**Number**			**x**		
THICKNESS	**Number**			**x**		
THICKNESS_UOM	**Varchar2(3)**			**x**		
WIDTH	Number			**x**		
WIDTH_UOM	**Varchar2(3)**			**x**		
CURL_WRINKLE_FOLD	**Varchar2(150)**			**x**		
LOT_ATTRIBUTE_CATEGORY	**Varchar2(30)**			**x**		
C_ATTRIBUTE1	**Varchar2(150)**			**x**		
C_ATTRIBUTE2	**Varchar2(150)**			**x**		
C_ATTRIBUTE3	**Varchar2(150)**			**x**		
C_ATTRIBUTE4	**Varchar2(150)**			**x**		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
C_ATTRIBUTE5	Varchar2(150)			x		
C_ATTRIBUTE6	Varchar2(150)			x		
C_ATTRIBUTE7	Varchar2(150)			x		
C_ATTRIBUTE8	Varchar2(150)			x		
C_ATTRIBUTE9	Varchar2(150)			x		
C_ATTRIBUTE10	Varchar2(150)			x		
C_ATTRIBUTE11	Varchar2(150)			x		
C_ATTRIBUTE12	Varchar2(150)			x		
C_ATTRIBUTE13	Varchar2(150)			x		
C_ATTRIBUTE14	Varchar2(150)			x		
C_ATTRIBUTE15	Varchar2(150)			x		
C_ATTRIBUTE16	Varchar2(150)			x		
C_ATTRIBUTE17	Varchar2(150)			x		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
C_ATTRIBUTE18	Varchar2(150)			x		
C_ATTRIBUTE19	Varchar2(150)			x		
C_ATTRIBUTE20	Varchar2(150)			x		
D_ATTRIBUTE1	Date			x		
D_ATTRIBUTE2	Date			x		
D_ATTRIBUTE3	Date			x		
D_ATTRIBUTE4	Date			x		
D_ATTRIBUTE5	Date			x		
D_ATTRIBUTE6	Date			x		
D_ATTRIBUTE7	Date			x		
D_ATTRIBUTE8	Date			x		
D_ATTRIBUTE9	Date			x		
D_ATTRIBUTE10	Date			x		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
N_ATTRIBUTE1	Number			x		
N_ATTRIBUTE2	Number			x		
N_ATTRIBUTE3	Number			x		
N_ATTRIBUTE4	Number			x		
N_ATTRIBUTE5	Number			x		
N_ATTRIBUTE6	Number			x		
N_ATTRIBUTE7	Number			x		
N_ATTRIBUTE8	Number			x		
N_ATTRIBUTE9	Number			x		
N_ATTRIBUTE10	Number			x		
VENDOR_NAME	Varchar2(240)			x		
SECONDARY_TRANSACTION_QUANTITY	Number			x		
SUBLOT_NUM	Varchar2(32)			x		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
REASON_CODE	Varchar2(4)			x		
ATTRIBUTE_CATEGORY	Varchar2(30)			x		
ATTRIBUTE1	Varchar2(150)			x		
ATTRIBUTE2	Varchar2(150)			x		
ATTRIBUTE3	Varchar2(150)			x		
ATTRIBUTE4	Varchar2(150)			x		
ATTRIBUTE5	Varchar2(150)			x		
ATTRIBUTE6	Varchar2(150)			x		
ATTRIBUTE7	Varchar2(150)			x		
ATTRIBUTE8	Varchar2(150)			x		
ATTRIBUTE9	Varchar2(150)			x		
ATTRIBUTE10	Varchar2(150)			x		
ATTRIBUTE11	Varchar2(150)			x		

Column Name	Data Type	Required	Derived	Optional	Receipt Type	Source Information
ATTRIBUTE12	Varchar2(150)			x		
ATTRIBUTE13	Varchar2(150)			x		
ATTRIBUTE14	Varchar2(150)			x		
ATTRIBUTE15	Varchar2(150)			x		

The values in the Required field indicate the following:

- 1- If item is under lot expiration control
- 2- If item is under both lot and serial control

LOT_NUMBER

Enter the lot number that is being transacted.

TRANSACTION_INTERFACE_ID

Use this column to associate lot transaction detail rows with the parent transaction row in MTL_TRANSACTIONS_INTERFACE.

SERIAL_TRANSACTION_TEMP_ID

This column is required only for items under both lot and serial control. It is used to identify the child rows in MTL_SERIAL_NUMBERS_INTERFACE.

PRODUCT_CODE

This column stores the product name to identify the source transaction for the record.

PRODUCT_TRANSACTION_ID

This column stores the identifier for the transaction in the table owned by the product specified under PRODUCT_CODE column

PROCESS_FLAG

Row Process Flag.

DESCRIPTION

This column is used only by Oracle Warehouse Management.

VENDOR_ID

use this column as identification for vendor. It is used only by Oracle Warehouse Management.

SUPPLIER_LOT_NUMBER

Use this column to list the supplier lot number. Its is used only by Oracle Warehouse Management.

TERRITORY_CODE

Use this column to specify the territory code for country of origin

ORIGINATION_DATE

Use this column to specify the origination date of the lot. It is used only by Oracle Warehouse Management.

DATE_CODE

Use this column to specify the code identifying the date of packaging or assembly. It is used only by Oracle Warehouse Management.

GRADE_CODE

Use this column for sublabeling of items, lots and sublots to identify their particular makeup and to separate one lot from other production lots of the same item. It is used only by Oracle Warehouse Management.

CHANGE_DATE

Use this column to enter attribute that defines the date on which an attribute of the lot or sublot was last changed. It is used only by Oracle Warehouse Management.

MATURITY_DATE

Use this column to enter the date on which the lot matures. It is used only by Oracle Warehouse Management.

STATUS_ID

Use this column to enter the identifier for the status of lot. It is used only by Oracle Warehouse Management.

RETEST_DAT

Restest Date. It is used only by Oracle Warehouse Management.

AGE

Use thsi colum to enter the age of the lot or sublot in days after the creation date.It is used only by Oracle Warehouse Management.

ITEM_SIZE

Use this column to enter the item size. It is used only by Oracle Warehouse Management.

COLOR

Color. It is used only by Oracle Warehouse Management.

VOLUME

Volume. It is used only by Oracle Warehouse Management.

VOLUME_UOM

Use this column to enter the UOM for measuring volume. It is used only by Oracle Warehouse Management.

PLACE_OF_ORIGIN

Use this column to enter the value indicating the place the inventory originated. It is used only by Oracle Warehouse Management.

BEST_BY_DATE

Use this column to enter the date on or before which the lot or sublot is best used. It is used only by Oracle Warehouse Management.

LENGTH

Length. It is used only by Oracle Warehouse Management.

LENGTH_UOM

Use this column to enter the UOM to measure length. It is used only by Oracle

Warehouse Management.

RECYCLED_CONTENT

The content, usually expressed as a percentage, of the product that is made up of recycled materials. It is used only by Oracle Warehouse Management.

THICKNESS

Thickness. It is used only by Oracle Warehouse Management.

THICKNESS_UOM

UOM to measure thickness. It is used only by Oracle Warehouse Management.

WIDTH

Width. It is used only by Oracle Warehouse Management.

WIDTH_UOM

UOM to measure width. It is used only by Oracle Warehouse Management.

CURL_WRINKLE_FOLD

This attribute is used in the pulp and paper industry to identify potential problems when feeding the paper through machinery. It is used only by Oracle Warehouse Management.

LOT_ATTRIBUTE_CATEGORY

Use this column to define the Lot attributes descriptive flexfield structure. It is used only by Oracle Warehouse Management.

C_ATTRIBUTE1- C_ATTRIBUTE20

Descriptive flexfield segment. It is used only by Oracle Warehouse Management.

D_ATTRIBUTE1- D_ATTRIBUTE10

Descriptive flexfield segment. It is used only by Oracle Warehouse Management.

N_ATTRIBUTE1- N_ATTRIBUTE10

Descriptive flexfield segment. It is used only by Oracle Warehouse Management.

VENDOR_NAME

Vendor name. It is used only by Oracle Warehouse Management.

SECONDARY_TRANSACTION_QUANTITY

Stores the secondary transaction quantity. It is used only by Oracle Warehouse Management.

SUBLOT_NUM

Sub-lot number.

REASON_CODE

Reason Code.

Attributes 1- 15

Descriptive Flexfield segment.

Serial Numbers Interface

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
TRANSACTION_INTERFACE_ID	Number	x			**PO, ASN, REQ, INV, RMA**	**MTL_MATERIAL_TRANSACTIONS_S**
SOURCE_CODE	Varchar2(30)			x		
FM_SERIAL_NUMBER	Varchar2(30)	x			**PO, ASN, REQ, INV, RMA**	
TO_SERIAL_NUMBER	Varchar2(30)			x	**PO, ASN, REQ, INV, RMA**	
SOURCE_LINE_ID	Number			x		
VENDOR_SERIAL_NUMBER	Varchar2(30)			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
ERROR_CODE	Varchar2(240)		x			
LAST_UPDATE_DATE	Date	x			**PO, ASN, REQ, INV, RMA**	**SYSDATE**
LAST_UPDATED_BY	Number	x			**PO, ASN, REQ, INV, RMA**	**FND_GLOBAL.USER_ID**
CREATION_DATE	Date	x			**PO, ASN, REQ, INV, RMA**	**SYSDATE**
CREATED_BY	Number	x			**PO, ASN, REQ, INV, RMA**	**FND_GLOBAL.USER_ID**
LAST_UPDATE_LOGIN	Number			x	**PO, ASN, REQ, INV, RMA**	**FND_GLOBAL.LOGIN_ID**
REQUEST_ID	Number			x		
PROGRAM_APPLICATION_ID	Number			x		
PROGRAM_ID	Number			x		
PROGRAM_UPDATE_DATE	Date			x		
PRODUCT_CODE	**Varchar2(5)**	**x**			**PO, ASN, REQ, INV, RMA**	**'RCV'**

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
PRODUCT_TRANSACTION_ID	Number	x			PO, ASN, REQ, INV, RMA	RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
VENDOR_LOT_NUMBER	Varchar2(30)			x		
PROCESS_FLAG	Number			x		
PARENT_SERIAL_NUMBER	Varchar2(30)			x		
SERIAL_ATTRIBUTE_CATEGORY	Varchar2(30)			x		
TERRITORY_CODE	Varchar2(30)			x		
ORIGINATION_DATE	Date			x		
C_ATTRIBUTE1	Varchar2(150)			x		
C_ATTRIBUTE2	Varchar2(150)			x		
C_ATTRIBUTE3	Varchar2(150)			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
C_ATTRIBUTE4	Varchar2(150)			x		
C_ATTRIBUTE5	Varchar2(150)			x		
C_ATTRIBUTE6	Varchar2(150)			x		
C_ATTRIBUTE7	Varchar2(150)			x		
C_ATTRIBUTE8	Varchar2(150)			x		
C_ATTRIBUTE9	Varchar2(150)			x		
C_ATTRIBUTE10	Varchar2(150)			x		
C_ATTRIBUTE11	Varchar2(150)			x		
C_ATTRIBUTE12	Varchar2(150)			x		
C_ATTRIBUTE13	Varchar2(150)			x		
C_ATTRIBUTE14	Varchar2(150)			x		
C_ATTRIBUTE15	Varchar2(150)			x		
C_ATTRIBUTE16	Varchar2(150)			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
C_ATTRIBUTE17	Varchar2(150)			x		
C_ATTRIBUTE18	Varchar2(150)			x		
C_ATTRIBUTE19	Varchar2(150)			x		
C_ATTRIBUTE20	Varchar2(150)			x		
D_ATTRIBUTE1	Date			x		
D_ATTRIBUTE2	Date			x		
D_ATTRIBUTE3	Date			x		
D_ATTRIBUTE4	Date			x		
D_ATTRIBUTE5	Date			x		
D_ATTRIBUTE6	Date			x		
D_ATTRIBUTE7	Date			x		
D_ATTRIBUTE8	Date			x		
D_ATTRIBUTE9	Date			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
D_ATTRIBUTE10	Date			x		
N_ATTRIBUTE1	Number			x		
N_ATTRIBUTE2	Number			x		
N_ATTRIBUTE3	Number			x		
N_ATTRIBUTE4	Number			x		
N_ATTRIBUTE5	Number			x		
N_ATTRIBUTE6	Number			x		
N_ATTRIBUTE7	Number			x		
N_ATTRIBUTE8	Number			x		
N_ATTRIBUTE9	Number			x		
N_ATTRIBUTE10	Number			x		
STATUS_ID	Number			x		
CYCLES_SINCE_NEW	Number			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
TIME_SINCE_OVERHAUL	Number			x		
CYCLES_SINCE_OVERHAUL	Number			x		
TIME_SINCE_REPAIR	Number			x		
CYCLES_SINCE_REPAIR	Number			x		
TIME_SINCE_VISIT	Number			x		
CYCLES_SINCE_VISIT	Number			x		
TIME_SINCE_MARK	Number			x		
CYCLES_SINCE_MARK	Number			x		
NUMBER_OF_REPAIRS	Number			x		
STATUS_NAME	Varchar2(30)			x		
C_ATTRIBUTE_17	Varchar2(150)			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
ATTRIBUTE_CATEGORY	Varchar2(30)			x		
ATTRIBUTE1	Varchar2(150)			x		
ATTRIBUTE2	Varchar2(150)			x		
ATTRIBUTE3	Varchar2(150)			x		
ATTRIBUTE4	Varchar2(150)			x		
ATTRIBUTE5	Varchar2(150)			x		
ATTRIBUTE6	Varchar2(150)			x		
ATTRIBUTE7	Varchar2(150)			x		
ATTRIBUTE8	Varchar2(150)			x		
ATTRIBUTE9	Varchar2(150)			x		
ATTRIBUTE10	Varchar2(150)			x		
ATTRIBUTE11	Varchar2(150)			x		
ATTRIBUTE12	Varchar2(150)			x		

Column Name	Type	Required	Derived	Optional	Receipt Type	Source Information
ATTRIBUTE13	**Varchar2(150)**			**x**		
ATTRIBUTE14	**Varchar2(150)**			**x**		
ATTRIBUTE15	**Varchar2(150)**			**x**		

FM_SERIAL_NUMBER

Enter the starting serial number in the range. If you enter only the 'from' serial number, the Transaction Processor assumes that only one serial number is being transacted.

TO_SERIAL_NUMBER

You can enter a 'to' serial number to specify a range. The transaction processor will attempt to transact all serial numbers within the range of the right-most numeric digits.

TRANSACTION_INTERFACE_ID

Use this column to associate serial number transaction detail rows with their parent rows. If the item is under both lot and serial control, this should point to MTL_TRANSACTION_LOTS_INTERFACE SERIAL_TRANSACTION_TEMP_ID. Otherwise, it should point to MTL_TRANSACTIONS_INTERFACE. TRANSACTION_INTERFACE_ID

VENDOR_SERIAL_NUMBER

You can use this column to enter vendor cross-reference information. The vendor serial number is stored in the serial number table MTL_SERIAL_NUMBERS.

PRODUCT_CODE

Stores the product name to identify the source transaction for the record.

PRODUCT_TRANSACTION_ID

Stores the identifier for the transaction in the table owned by the product specified under PRODUCT_CODE column.

VENDOR_LOT_NUMBER

Supplier lot number.

PROCESS_FLAG

Row process flag .

PARENT_SERIAL_NUMBER

Parent serial number.

SERIAL_ATTRIBUTE_CATEGORY

Use this column to define structure for Serial Attributes flex field. It is used only by Oracle Warehouse Management.

TERRITORY_CODE

Territory code for Country of Origin. It is used only by Oracle Warehouse Management.

ORIGINATION_DATE

Creation Date of the serial number such as manufacture date. It is used only by Oracle Warehouse Management.

C_ATTRIBUTE1- C_ATTRIBUTE20

Descriptive flexfield segment. It is used only by Oracle Warehouse Management.

D_ATTRIBUTE1- D_ATTRIBUTE10

Descriptive flexfield segment. It is used only by Oracle Warehouse Management.

N_ATTRIBUTE1- N_ATTRIBUTE10

Descriptive flexfield segment. It is used only by Oracle Warehouse Management.

STATUS_ID

Status identifier. It is used only by Oracle Warehouse Management.

CYCLES_SINCE_NEW

Cycles since new for MRO industry. It is used only by Oracle Warehouse Management.

TIME_SINCE_OVERHAUL

Time Since overhaul for MRO industry. It is used only by Oracle Warehouse

Management.

CYCLES_SINCE_OVERHAUL

Cycles since overhaul for MRO industry. It is used only by Oracle Warehouse Management.

TIME_SINCE_REPAIR

Time since repair. It is used only by Oracle Warehouse Management.

CYCLES_SINCE_REPAIR

Cycles since repair. It is used only by Oracle Warehouse Management.

TIME_SINCE_VISIT

Time since visit. It is used only by Oracle Warehouse Management.

CYCLES_SINCE_VISIT

Cycles since visit. It is used only by Oracle Warehouse Management.

TIME_SINCE_MARK

Time since the serial Number is marked. It is used only by Oracle Warehouse Management.

CYCLES_SINCE_MARK

Number of cycles since the serial number was marked. It is used only by Oracle Warehouse Management.

NUMBER_OF_REPAIRS

Number of repairs. It is used only by Oracle Warehouse Management.

STATUS_NAME

Status of the serial number.

ATTRIBUTE_CATEGORY

Descriptive flexfield segment.

ATTRIBUTE1- ATTRIBUTE15

Descriptive flexfield segment.

CST_COMP_SNAP_INTERFACE

Column Name	Type	Required	Derived	Optional
TRANSACTION_INTERFACE_ID	Number	x		
WIP_ENTITY_ID	Number	x		
OPERATION_SEQ_NUMBER	Number	x		
LAST_UPDATE_DATE	Date	x		
LAST_UPDATED_BY	Number	x		
CREATION_DATE	Date	x		
CREATED_BY	Number	x		
LAST_UPDATE_LOGIN	Number			x
NEW_OPERATION_FLAG			x	
PRIMARY_QUANTITY			x	
QUANTITY_COMPLETED	Number	x		
PRIOR_COMPLETION_QUANTITY			x	
PRIOR_SCRAP_QUANTITY			x	

Column Name	Type	Required	Derived	Optional
REQUEST_ID	Number			x
PROGRAM_APPLICATION_ID	Number			x
PROGRAM_ID	Number			x
PROGRAM_UPDATE_DATE	Date			x

WIP_ENTITY_ID

The job number.

OPERATION_SEQ_NUMBER

You can use this column to enter operation sequence information. The operation sequence number is stored in the WIP operations table WIP_OPERATIONS.

Validation

Oracle Inventory lets you choose the level of validation you want performed against interfaced transaction rows. Using the VALIDATION_REQUIRED flag, you can specify whether you want full validation or only partial validation of columns required for derivation of other required columns. For example, ORGANIZATION_ID is always validated because there are dependent attributes such as LOCATOR_ID that require a valid organization for derivation. REVISION, on the other hand, has no dependencies, and therefore is not validated if the VALIDATION_REQUIRED flag is not set.

The validation and derivation processes will provide an error code and description for all transaction rows that fail explicit validation checks. If an error occurs during reservation relief for a specific transaction, all rows in the transaction processing group will be errored out with a common error message. This should happen, however, only if there is an Oracle error or table deadlock during processing.

If an error occurs in the transaction processor, the entire transaction processing group is marked with the error code, while the transaction row(s) that actually failed will have an error explanation.

Resolving Failed Transaction Interface Rows

Viewing Failed Transactions

You can view both pending and failed Inventory transactions in the MTL_TRANSACTIONS_INTERFACE table using the Pending Transactions window. If your transactions errored out and you would like to resubmit them, you can do so using this window. If you set 'Resubmit=Yes', the interface processing flags will automatically be reset so the Transaction Manager will pick them up. See: Viewing Pending Transactions, *Oracle Inventory User's Guide*.

Fixing Failed Transactions Options

Errors in the interface may be caused by problems unrelated to your interfaced transactions. For example, there may be validation that failed because an entity that was being checked had the wrong status (for example, disabled), or the failure could even be the result of a system error, such as running out of space. In these cases, it may be acceptable to simply resolve the conflict and resubmit the same interfaced rows by either using the Pending Transactions window to resubmit your transactions, or by directly updating the PROCESS_FLAG and LOCK_FLAG values via SQL*PLUS.

If, however, you need to make changes to the transaction data itself, you need to either delete the failed transactions and resubmit them from the feeder system, or update the transaction in the interface table using SQL*PLUS. When you resubmit updated transactions for processing, all validation is performed again.

Transaction Flow Application Program Interface

Transaction Flows define the sequence of operating units that need to have inter-company accounting in order fulfillment, procuring or drop shipment flows. Each operating unit pair in this sequence is subjected to inter-company accounting.

A transaction flow is made up of a header for which there is one or more transaction flow lines. Each line has an inter-company setup.

The Transaction Flow API allows you to create transaction flows.

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_data	OUT	Varchar2			
x_msg_count	OUT	Number			

Parameter	Usage	Type	Required	Derived	Optional
x_header_id	OUT	Number			
x_line_number_tbl	OUT	PL/SQL table			
p_api_version	IN	Number	X		
p_init_msg_list	IN	Varchar2		X	X
p_validation_level	IN	Number	X		
p_start_org_id	IN	Number	X		
p_end_org_id	IN	Number	X		
p_flow_type	IN	Number	X		
p_organization_id	IN	Number			X
p_qualifier_code	IN	Number			X
p_qualifier_value_id	IN	Number			X
p_asset_item_pricing_option	IN	Number	X		
p_expense_item_pricing_option	IN	Number	X		
p_new_accounting_flag	IN	Varchar2	X		
p_start_date	IN	Date		X	X

Parameter	Usage	Type	Required	Derived	Optional
p_end_date	IN	Date			X
P_Attribute_ Category	IN	Varchar2			X
P_Attribute1 to P_Attribute15	IN	Varchar2			X
p_line_numb er_tbl	IN	PL/SQL table	X		
p_from_org_i d_tbl	IN	PL/SQL table	X		
p_from_orga nization_id_t bl	IN	PL/SQL table	X		
p_to_org_id_t bl	IN	PL/SQL table	X		
p_to_organiz ation_id_tbl	IN	PL/SQL table	X		
P_LINE_Attri bute_Categor y_tbl	IN	PL/SQL table	X		
P_LINE_Attri bute1_tbl – P_LINE_Attri bute15_tbl	IN	PL/SQL table	X		
P_Ship_Orga nization_Id_t bl	IN	PL/SQL table	X		
P_Sell_Organ ization_Id_tbl	IN	PL/SQL Table	X		

Parameter	Usage	Type	Required	Derived	Optional
P_Vendor_Id_tbl	IN	PL/SQL table	X		
P_Vendor_Site_Id_tbl	IN	PL/SQL table	X		
P_Customer_Id_tbl	IN	PL/SQL table	X		
P_Address_Id_tbl	IN	PL/SQL table	X		
P_Customer_Site_Id_tbl	IN	PL/SQL table	X		
P_Cust_Trx_Type_Id_tbl	IN	PL/SQL table	X		
P_IC_Attribute_Category_tbl	IN	PL/SQL table	X		
P_IC_Attribute1_tbl – P_IC_Attribute15_tbl	IN	PL/SQL table	X		
P_Revalue_Average_Flag_tbl	IN	PL/SQL table	X		
P_Freight_Code_Comb_Id_tbl	IN	PL/SQL table	X		
P_Inv_Currency_Code_tbl	IN	PL/SQL table	X		
P_IC_COGS_Acct_Id_tbl	IN	PL/SQL table	X		

Parameter	Usage	Type	Required	Derived	Optional
P_Inv_Accrual_Acct_Id_tbl	IN	PL/SQL table	X		
P_Exp_Accrual_Acct_Id_tbl	IN	PL/SQL table	X		

x_return_status

Return status of the API. Valid values include:Success: FND_API.G_RET_STS_SUCCESSError: FND_API.G_RET_STS_EXC_ERRORUnexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If x_msg_count is equal to 1 then this contains the actual message.

x_header_id

Header Id of the transaction flow created.

x_line_number_tbl

Table of line umbers of transaction flow lines created.

p_api_version

Indicates the API version. For now it is 1.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- _ p_msg_index => I

- _ p_encoded => F
- _ p_data => 1_message
- _ p_msg_index_out => 1_msg_index_out
- where 1_message and 1_msg_index_out are local variables of types
- Varchar2(2000 and Number respectively.
- Default Value: FND_API.G_FALSE

p_validation_level

Indicate fnd_api.g_valid_level_full to enable validation.

p_start_org_id

The starting operating unit for the transaction flow to be defined. For a shipment flow, it is the Shipping operating unit. For a procuring flow, it is the Procuring operating unit.

p_end_org_id

The ending operating unit for the transaction flow to be defined. For a shipment flow, it is the Selling operating unit. For a procuring flow, it is the Receiving operating unit.

p_flow_type

1. Shipping flow
2. Procuring flow

p_organization_id

An optional parameter that indicates the shipping organization if the transaction flow is shipping or receiving organization if the transaction flow is procuring.

p_qualifier_code

An optional parameter that indicates the qualifier type for which the transaction flow is created. For now only one value (1- category) is supported.

p_qualifier_value_id

For p_qualifier_code of 1, this indicates a valid category in 'Inventory' category set for shipping flows and a valid category in 'Purchasing' category set for procuring flows.

p_asset_item_pricing_option

For procuring flows indicates the pricing mechanism for asset items

PO price

1. Transfer price

p_expense_item_pricing_option

For procuring flows indicates the pricing mechanism for expense items

1. PO price
2. Transfer price

p_new_accounting _flag

- For procuring flows this will always have to be 'Y'.
  - For shipping flows with more than 2 operating units, this will always have to be 'Y'.
  - For other cases this can be 'Y' or 'N'. 'N' indicates the accounting prior to release 11i10.

p_start_date

The starting date for which this transaction flow will be effective.

p_end_date

The ending date for which this transaction flow will be effective. A value of null indicates it is effective till end of time.

p_attribute_category and p_attribute1 to p_attribute15

Descriptive flex information for transaction flow header.

p_line_number_tbl

Table of line numbers for the transaction flow lines.

p_from_org_id_tbl

The from-operating unit in each transaction flow line.

p_from_organization_id_tbl

A default inventory organization for the from-operating unit in each transaction flow line.

p_to_org_id_tbl

The to-operating unit in each transaction flow line.

p_to_organization_id_tbl

A default inventory organization for the to-operating unit in each transaction flow line.

P_LINE_Attribute_Category_tbl and P_LINE_Attribute1_tbl to P_LINE_Attribute15_tbl

Descriptive flex information for each of the transaction flow lines.

P_Ship_Organization_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows, this is the Shipping operating unit.
- For Procuring flows, this is the Procuring operating unit.

P_Sell_Organization_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows with Sales Orders, this is the Selling operating unit.
- For Shipping flows with Internal Orders, this is the Receiving operating unit.
- For Procuring flows, this is the Receiving operating unit.

P_Vendor_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows, this is the internal vendor associated with Shipping operating unit.
- For Procuring flows, this is the internal vendor associated with Procuring operating unit.

P_Vendor_Site_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows, this is the internal vendor site associated with Shipping operating unit.
- For Procuring flows, this is the internal vendor site associated with Procuring operating unit.

P_Customer_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows with Sales Orders, this is the internal customer associated with Selling operating unit.
- For Shipping flows with Internal Orders, this is the internal customer associated with Receiving operating unit.
- For Procuring flows, this is the internal customer associated with Receiving operating unit.

P_Address_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows with Sales Orders, this is the address associated with Selling operating unit.
- For Shipping flows with Internal Orders, this is the address associated with Receiving operating unit.
- For Procuring flows, this is the address associated with Receiving operating unit.

P_Customer_Site_Id_tbl

Defined for each transaction flow line as:

- For Shipping flows with Sales Orders, this is the internal customer site associated with Selling operating unit.
- For Shipping flows with Internal Orders, this is the internal customer site associated with Receiving operating unit. For Procuring flows, this is the internal customer site associated with Receiving operating unit.

P_Cust_Trx_Type_Id_tbl

Defined for each transaction flow line this is the transaction type to be used for Inter-Company AR/AP.

P_IC_Attribute_Category_tbl and P_IC_Attribute1_tbl to P_IC_Attribute15_tbl

Descriptive flex information for inter-company relation mapping to each transaction flow line.

P_Revalue_Average_Flag_tbl

This table can have null for each transaction flow line as this functionality is obsolete.

P_Freight_Code_Comb_Id_tbl

Freight Account to be used for inter-company AR in each transaction flow line.

P_Inv_Currency_Code_tbl

Currency to be used for inter-company AR in each transaction flow line.

1. Order Currency (Sales Order/Purchase Order)
2. Currency of the Ship/From/Procuring operating unit
3. Currency of the Sell/To/Receiving operating unit

P_IC_COGS_Acct_Id_tbl

COGS account to be used against Inter-Company sale in each transaction flow line.

P_Inv_Accrual_Acct_Id_tbl

Inter-Company Accrual account to be used for asset items in Inter-Company accounting in each transaction flow line.

P_Exp_Accrual_Acct_Id_tbl

Inter-Company Accrual account to be used for expense items in Inter-Company accounting in each transaction flow line.

Open Replenishment Interface

Oracle Inventory provides an open interface for you to easily load replenishment requests from external systems such as a barcode application. Such requests may be in the form of stock-take counts or requisition requests for subinventories in which you do not track quantities.

You may also use the Replenishment Interface to process requisition requests generated by external applications for tracked subinventories.

You must write the load program that inserts a single row for each replenishment count/request into the MTL_REPLENISH_HEADERS _INT table. A record for each item included in the count header must be inserted into the MTL_REPLENISH_LINES_INT table.

There are two modes you can use to send your replenishment counts through the interface. These are Concurrent and Background modes.

Under Concurrent mode processing, you populate the interface tables for a specific replenishment count and then call the replenishment validator from the Oracle Inventory menu (Counting/Replenishment Counts/Process Interface). The validator processes the replenishment count specified as a parameter at process submission, validating rows in both the MTL_REPLENISH_HEADER_INT and MTL_REPLENISH_LINES_INT tables. The validator derives any additional columns and updates the error flag if an error is detected.

For Background mode processing, you populate the interface tables and then let the Replenishment Validator asynchronously poll the tables for replenishment counts to process.

If the replenishment count, both header and lines, passes all required validation, the records are inserted into the MTL_REPLENISH_ HEADERS and MTL_REPLENISH_LINES tables and are deleted from the interface tables. If an error is detected during the validation process, the header and corresponding replenishment lines will be left in the interface table.

Once the lines are in the internal replenishment tables, you use the Replenishment Processor as described in the *Oracle Inventory User's Guide* to process the counts and create requisitions. See: Entering and Processing Replenishment Counts, *Oracle Inventory User's Guide*.

Access the Replenishment Interface through the Oracle Inventory menu (Counting/Replenishment Counts/Process Interface). Select the type of request by choosing Single Request. In the Request Name field, select Validate Replenishment Interface. In the Parameters window, select Concurrent or Background as the Processing Mode and select the Count Name for processing. Select Submit Request to begin processing. You can also use the Schedule button to specify resubmission parameters that will control how frequently the Replenishment Validator polls for records in the interface tables.

Inserting into the Replenishment Interface Tables

This section provides a chart for each interface table that lists all columns, followed by a section giving a brief description of a subset of columns requiring further explanation. The chart identifies each column's datatype and whether it is Required, Derived, or Optional.

Several of the attributes in the interface tables can be populated using either the user-friendly values or the internal identifiers. For example, you have the choice of specifying either the flexfield segment representation or the internal identifier (e.g.

INVENTORY_ITEM_ID) for the required value. When specifying the organization, you may either use the organization code or the internal identifier (e.g. ORGANIZATION_ID).

If you populate the user friendly values, the Replenishment Validator will validate them and will derive the internal identifiers. If the translation is available to the external system, it may be advantageous to use the internal identifiers to improve performance.

Replenishment Headers Interface Tables

The following graphic describes the MTL_REPLENISH_HEADERS_INT table:

Column Name	Type	Required	Derived	Optional
REPLENISHMENT_COUNT_NAME	Varchar2(10)	3		
COUNT_DATE	Date	3		
LAST_UPDATE_DATE	Date	3		
CREATION_DATE	Date	3		
CREATED_BY	Number	3		
LAST_UPDATE_LOGIN	Number			3
LAST_UPDATED_BY	Number	3		
ORGANIZATION_ID	Number	3		
ORGANIZATION_CODE	Varchar2(3)	3		
SUBINVENTORY_CODE	Varchar2(10)	3		
SUPPLY_CUTOFF_DATE	Date		x	3

Column Name	Type	Required	Derived	Optional
PROCESS_STATUS	Number	3		
PROCESS_MODE	Number	3		
ERROR_FLAG	Number		3	
REQUEST_ID	Number		3	
PROGRAM_APPLICATION_ID	Number		3	
PROGRAM_ID	Number		3	
PROGRAM_UPDATE_DATE	Date		3	
DELIVERY_LOCATION_ID	Number			3
DELIVERY_LOCATION_CODE	Varchar2(20)			3
LOCATOR_ID	Number			
LOCATOR_SEGMENT1-20	Varchar2(40)			

ERROR_FLAG

If a validation error occurs, the replenishment validator populates this column with an error code. The error flag for a replenishment header will be set if either the validation of the header fails or if the validation of any of the lines of the header fails.

ORGANIZATION_ID

This column identifies the internal identifier of the organization from which the replenishment count originated. You must enter either the internal organization identifier or the user friendly organization code.

ORGANIZATION_CODE

This column is the user friendly code for the organization that is the source of the replenishment count. It may be used instead of the internal identifier, in which case the internal identifier will be derived.

PROCESS_MODE

This column determines how the interfaced replenishment count will be processed. The valid options are:

2 - Concurrent

3 - Background

Interface replenishment counts marked for Background processing will be picked up by the replenishment validator polling process. The validator will pick up and process all replenishment counts with a process mode of Background each time it runs.

You use Concurrent processing mode if you want to launch a dedicated replenishment validator process to explicitly process a single replenishment count, identified as a parameter to the program, from the interface table.

PROCESS_STATUS

This column is used to identify the current processing status of the replenishment count. You should insert rows that you intend to be processed with a value of 2 (Pending). The valid values for this column are:

1. Pending
2. Processing
3. Error
4. Completed

Hold

If you want to insert records into the interface tables but temporarily prevent them from being processed, you can use this column by setting the value to 1 (Hold).

After the validator has run, it will set the value of this column to 5 (Completed). This status is used whenever the process completes, whether validation errors were detected or not.

A status of 4 (Error) indicates an internal error occurred. This error indicates an exceptional condition and should not occur.

REPLENISH_COUNT_NAME

Enter a unique name for the replenishment count.

SUBINVENTORY_CODE

This column identifies the subinventory that is the source of the replenishment count.

SUPPLY_CUTOFF_DATE

Enter the date after which planned supply will not be considered in available quantity calculations. The value for this column will be derived from system date if it is left null.

DELIVERY_LOCATION_ID

Enter the internal identifier for the location to which the replenishment should be delivered. You may enter the delivery location identifier, the user friendly delivery location code or neither. If neither is specified, the default delivery location for the organization from which the replenishment originated is defaulted.

DELIVERY_LOCATION_CODE

Enter the user friendly code for the delivery location of the replenishment. You may enter this code instead of the internal identifier, in which case the internal identifier will be derived. You may specify neither the code or the identifier, in which case the default delivery location of the organization originating the replenishment will be used.

The following graphic describes the MTL_REPLENISH_LINES_INT table:

Column Name	Type	Required	Derived	Optional
REPLENISHMENT_HEADER_ID	Number	3		
REPLENISHMENT_LINE_ID	Number	3		
ORGANIZATION_ID	Number			3
LAST_UPDATE_DATE	Date	3		
CREATION_DATE	Date	3		
CREATED_BY	Number	3		

Column Name	Type	Required	Derived	Optional
LAST_UPDATE_LOGIN	Number	3		
LAST_UPDATED_BY	Number	3		
INVENTORY_ITEM_ID	Number	3		
SEGMENT {1-20}	Varchar2(40)	3		
COUNT_TYPE_CODE	Number	3		
COUNT_QUANTITY	Number	3		
REFERENCE	Varchar2(240)			3
ERROR_FLAG	Number		3	
REQUEST_ID	Number		3	
PROGRAM_APPLICATION_ID	Number		3	
PROGRAM_ID	Number		3	
PROGRAM_UPDATE_DATE	Date		3	
COUNT_UNIT_OF_MEASURE	Varchar2(25)	3		
COUNT_UOM_CODE	Varchar2(3)	3		

REPLENISHMENT_HEADER_ID

Enter the unique identifier of the replenishment count. The identifier entered here is the foreign key reference which links the header table with the lines table to associate a group of lines with a single header.

REPLENISHMENT_LINE_ID

Enter the identifier for the line within the replenishment count. You may use the sequence MTL_REPLENISH_LINES_S to obtain a unique identifier for the line.

INVENTORY_ITEM_ID

Enter the internal identifier for the item to be replenished.

SEGMENT{1-20}

You may use these flexfield columns instead of INVENTORY_ITEM_ID to enter the item identifier in a more user-friendly form.

ORGANIZATION_ID

This column identifies the internal identifier of the organization from which the replenishment count originated. If you do not enter a value here, the organization identifier will be derived from the replenishment header.

COUNT_TYPE_CODE

Enter the type of the replenishment count entry. The valid count types are:

1. On-hand Quantity
2. Order Quantity
3. Order Maximum

Use On-hand Quantity to identify counts that are the result of stock-takes of subinventories in which you do not track on-hand quantities.

Use Order Quantity when you want to specify the quantity to be ordered. This count type may be used with either tracked or non-tracked subinventories.

Use Order Maximum when you want to place an order for the min-max maximum quantity specified for item in the subinventory specified. This count type may be used with either tracked or non-tracked subinventories.

COUNT_QUANTITY

This column is used to specify the count quantity that corresponds to the count type entered for the line. When the count type is On-hand Quantity, the count quantity is the on-hand balance determined during the stock-take. When the count type is Order Quantity, the count quantity represents the quantity to be ordered. This column is not used when the count type is Order Maximum.

REFERENCE

Use this column to enter any replenishment count reference information.

COUNT_UNIT_OF_MEASURE

Enter the count unit of measure identifier. This column may be used to specify the full name for the unit of measure. This column is meaningful only when a value is entered in the COUNT_QUANTITY columns.

COUNT_UOM_CODE

This column is represents the unit of measure code used for the count. You may specify the code when populating this table or you may use the full name for the unit of measure, in which case this column will be derived. This column is meaningful only when a value is entered in the COUNT_QUANTITY columns.

ERROR_FLAG

This flag indicates the error status of the validation of a replenishment line. The replenishment validator populates this column with a line corresponding to the error detected during validation.

Validation

Oracle Inventory validates the following conditions:

- The value of REPLENISH_HEADER_ID must be unique among existing replenishment counts
- The value of REPLENISH_COUNT_NAME must be unique among existing count headers
- The value of LAST_UPDATED_BY must be a valid user name
- ORGANIZATION_ID must be a valid identifier of an organization
- SUBINVENTORY_CODE must refer to an existing subinventory
- DELIVERY_LOCATION_ID must be a valid identifier of a location associated with the organization generating the replenishment
- There must be at least one line per header
- The ORGANIZATION_ID at the header level must be the same as that at the line level
- COUNT_TYPE_CODE must be either 1, 2, or 3 and must be consistent with whether

the subinventory is tracked or non-tracked

- The value of COUNT_QUANTITY must be consistent with COUNT_TYPE_CODE and must be greater than zero
- INVENTORY_ITEM_ID must refer to a transactable item in the organization specified
- The item must exist in the subinventory and must be min-max planned in that subinventory
- The COUNT_UOM_CODE must be valid and conversions to primary UOM must exist
- Each line must correspond to a header

Viewing Failed Transactions

Replenishment counts that fail the validation process will remain in the MTL_REPLENISH_HEADERS_INT and MTL_REPLENISH_LINES_INT tables. You may use SQL*PLUS to identify the headers that have failed by selecting those rows with a process_status of 5 (Complete). The reason for the failure will be reflected in the ERROR_FLAG column.

Possible values for the ERROR_FLAG column in the MTL_REPLENISH_HEADERS_INT table are:

1 - Non-unique replenishment header id

2 - Non-unique replenishment count name

3 - Invalid user name

4 - Invalid organization identifier

5 - Invalid subinventory

7 - Header with no corresponding replenishment lines

10 - Header failed because line failed

18 - Delivery location is not valid

Possible values for the ERROR_FLAG column in the MTL_REPLENISH_LINES_INT table are:

1 - No corresponding header id

3 - Invalid user name

8 - Invalid item identifier or item isn't transactable

9 - Invalid unit of measure or no conversion to primary unit of measure exists

11 - No item specified in either identifier or segments

12 - Invalid count type

13 - On-hand count type used for tracked subinventory

14 - Invalid count quantity

15 - Lines organization header does not match header organization identifier

17 - Item is not specified in the subinventory or is not min-max planned in the subinventory

Fixing Failed Transactions

Frequently, errors in the interface are caused by problems external to the replenishment count itself. For example, there may be validation that failed because an entity that was being validated had the wrong status (i.e. disabled), or the failure could even be the result of a system error, such as running out of space. In these cases, the resolution is simple; once you have made the necessary changes, you simply need to resubmit the replenishment validator process.

If, however, you need to make changes to the data in the interface table, you need to either delete the failed records, correct them in the external feeder system and resubmit them, or update the interface record in the interface table using SQL*PLUS. When you resubmit updated transactions for processing, all validation will be performed again.

Cycle Count Entries Interface

You can import cycle count entries from an external system into Oracle Inventory using the Cycle Count Entries Interface. This interface validates all data that you import into Oracle Inventory. It also performs foreign key validation and checks for attribute inter-dependencies, acceptable values, and value ranges. The interface ensures that the imported cycle count entries contain the same detail as items entered manually using the Cycle Count Entries window. Errors detected during validation are written to the Cycle Count Interface Errors table.

Interface Runtime Options

You can use the Concurrent Program "Import cycle count entries from open interface" to import the data from the Cycle Count Entries Interface table to the Cycle Count Entries table after validation. The program expects the following parameters:

- Cycle Count Name
- Number of Workers
- Commit Point
- Error Report Level

- Delete Successful Records

Cycle Count Entries Interface Table

Table Description

The Cycle Count Entries Interface table, MTL_CC_ENTRIES_INTERFACE, includes all the columns in the Cycle Count Entries table, MTL_CYCLE_COUNT_ENTRIES.

Note: Information about columns that need to be populated for audit trail and maintenance purposes can be found in the Table Administration and Audit Trail section. See Table Administration and Audit Trail.

Note: For information about columns not discussed in the Interface manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual*.

Field Name	Type	Null
CC_ENTRY_INTERFACE_ID	Number	N
ORGANIZATION_ID	Number	N
LAST_UPDATE_DATE	Date	N
LAST_UPDATED_BY	Number	N
CREATION_DATE	Date	N
CREATED_BY	Number	N
LAST_UPDATE_LOGIN	Number	Y
CC_ENTRY_INTERFACE_GROUP_ID	Number	Y
CYCLE_COUNT_ENTRY_ID	Number	Y
ACTION_CODE	Number	N

Field Name	Type	Null
CYCLE_COUNT_HEADER_ID	Number	Y
CYCLE_COUNT_HEADER_NAME	Varchar2(30)	Y
COUNT_LIST_SEQUENCE	Number	Y
INVENTORY_ITEM_ID	Number	Y
ITEM_SEGMENT1-20	Varchar2(40)	Y
REVISION	Varchar2(3)	Y
SUBINVENTORY	Varchar2(10)	Y
LOCATOR_ID	Number	Y
LOCATOR_SEGMENT1-20	Varchar2(40)	Y
LOT_NUMBER	Varchar2(30)	Y
SERIAL_NUMBER	Varchar2(30)	Y
PRIMARY_UOM_QUANTITY	Number	Y
COUNT_UOM	Varchar2(3)	Y
COUNT_UNIT_OF_MEASURE	Varchar2(25)	Y
COUNT_QUANTITY	Number	Y
SYSTEM_QUANTITY	Number	Y
ADJUSTMENT_ACCOUNT_ID	Number	Y
ACCOUNT_SEGMENT1-30	Varchar2(25)	Y

Field Name	Type	Null
COUNT_DATE	Date	Y
EMPLOYEE_ID	Number	Y
EMPLOYEE_FULL_NAME	Varchar2(240)	Y
REFERENCE	Varchar2(240)	
TRANSACTION_REASON_ID	Number	Y
TRANSACTION_REASON	Varchar2(30)	Y
REQUEST_ID	Number	Y
PROGRAM_APPLICATION_ID	Number	Y
PROGRAM_ID	Number	Y
PROGRAM_UPDATE_DATE	Date	Y
LOCK_FLAG	Number	Y
PROCESS_FLAG	Number	Y
PROCESS_MODE	Number	Y
VALID_FLAG	Number	Y
DELETE_FLAG	Number	Y
STATUS_FLAG	Number	Y
ERROR_FLAG	Number	Y
ATTRIBUTE_CATEGORY	Varchar2(30)	Y
ATTRIBUTE1-15	Varchar2(150)	Y

Field Name	Type	Null
PROJECT_ID	Number	Y
TASK_ID	Number	Y
SYSTEM_QUANTITY	Number	Y
PARENT_LPN_ID	Number	Y
OUTERMOST_LPN_ID	Number	Y
PARENT_LPN	VARCHAR2(30)	Y
COST_GROUP_ID	Number	Y
COST_GROUP_NAME	VARCHAR2(10)	Y
SECONDARY_COUNT_QUANTITY	Number	Y
SECONDARY_SYSTEM_QUANTITY	Number	Y
SECONDARY_UNIT_OF_MEASURE	VARCHAR2(25)	Y
SECONDARY_UOM	VARCHAR2(3)	Y

Cycle Count Interface Errors Table

Table Description

The Cycle Count Interface Errors table, MTL_CC_INTERFACE_ERRORS, is populated with errors encountered while processing interface rows. This table provides for the reporting of multiple errors for each interface record.

> **Note:** Information about columns that need to be populated for audit trail and maintenance purposes can be found in the Table Administration and Audit Trail section. See Cycle Count Entries Interface.

Note: For information about columns not discussed in the Interface manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual.*

Field Name	Type	Null
INTERFACE_ERROR_ID	Number	N
CC_ENTRY_INTERFACE_ID	Number	N
LAST_UPDATE_DATE	Date	N
LAST_UPDATED_BY	Number	N
CREATION_DATE	Date	N
CREATED_BY	Number	N
LAST_UPDATE_LOGIN	Number	Y
REQUEST_ID	Number	Y
PROGRAM_APPLICATION_ID	Number	Y
PROGRAM_ID	Number	Y
PROGRAM_UPDATE_DATE	Date	Y
ERROR_MESSAGE	Varchar2(240)	Y
ERROR_COLUMN_NAME	Varchar2(32)	Y
ERROR_TABLE_NAME	Varchar2(30)	Y
MESSAGE_NAME	Varchar2(30)	Y

Table Administration and Audit Trail

Some columns in the Interface tables are required for audit trail maintenance and table administration data. This section explains the purpose of these Standard Who columns.

- LAST_UPDATE_DATE should contain the date on which the record was last updated. Use the SQL function SYSDATE in this column to automatically record the current system date when the record is updated. This is a required field.
- LAST_UPDATED_BY field is populated with the user identification number of the person updating the customer item tables. Follow your organization's convention for the user identification number to populate this field. This is a required field.
- CREATION_DATE contains the date on which a particular customer item record was created. Populate this field according to your organization's convention if this information is not available from the legacy system. This is a required field.
- CREATED_BY contains the user identification number of the person who originally created this customer item record. Follow your organization's convention for generating user identification numbers to populate this field if this information is not available from the legacy system. This is a required field.
- LAST_UPDATE_LOGIN is not a required field. This field is currently not being used and should be populated with -1.
- REQUEST_ID is the concurrent request identifier of the last concurrent program to affect that record. This is not a required field and can be Null.
- PROGRAM_APPLICATION_ID is the application identifier of the owner of the program to last affect that record. This is not a required field and can be Null.
- PROGRAM_ID is the program identifier of the last record to affect the record. This is not a required field and can be Null.
- PROGRAM_UPDATE_DATE is the last date on which a program updated that record. This is not a required field and can be Null.

Cycle Count Application Program Interface

The Cycle Count API is a public API that allows you to perform on-line processing for cycle count records. This API takes a cycle count interface row that has been passed through the p_interface_rec parameter and processes it based on the values of the parameter fields. The Cycle Count API includes the public procedure import_countrequest.

Setting Up the Cycle Count API

Parameter Descriptions

The following chart lists all parameters used by the Cycle Count API. Additional information on these parameters follows the chart.

IMPORT_COUNTREQUEST

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2			x
p_commit	IN	Varchar2			x
p_validation_level	IN	Number			x
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_interface_rec	IN	Record	x		

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

p_validation_level

Default Value: FND_API.G_VALID_LEVEL_FULL

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_interface_rec

Indicates the complete interface record.

Validation of Cycle Count API

Standard Validation

Oracle Inventory validates all input parameters in the Cycle Count API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Cycle Count API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Kanban Application Program Interface

The Kanban API is a public API that allows you to update the supply status of kanban cards. To accomplish this task, you use the public procedure update_card_supply_status.

Setting Up the Kanban API

The following chart lists all parameters used by the update_card_supply_status procedure. Additional information on these parameters follows the chart.

UPDATE_CARD_SUPPLY_STATUS

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_level	IN	Varchar2		x	
p_commit	IN	Varchar2		x	
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
p_kanban_card_id	IN	Varchar2	x		
p_supply_status	IN	Varchar2	x		

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the

actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

p_kanban_card_id

Indicates the kanban card identifier to be updated.

p_supply_status

Indicates the updated status of the kanban card.

Validation of Kanban API

Standard Validation

Oracle Inventory validates all input parameters in the update_card_supply_status procedure. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Kanban API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Lot Application Program Interface

The Lot API has the following public procedures:

Create_Inv_lot

This procedure inserts a lot into the MTL_LOT_NUMBERS table. It does all the necessary validation before inserting the lot. It derives the expiration date depending on the controls set for shelf_life_code and shelf_life_days for the item. Returns success if it is able to insert the lot. If the lot already exists for the same item and org, it places a message on the stack informing that the lot already exists. It returns error if there is any validation error. Standard WHO information is used from the fnd_global api.

create_inv_lot

This procedure accepts the lot attributes as input parameters. This procedure validates some the input parameters like lot_number, expiration date, named attributes e.g. change date, age, retest date, C_ATTRIBUTES, N_ATTRIBUTES and D_ATTRIBUTES. If all the validations go through fine, new lot is created with the attribute information.

Auto_gen_lot

This function automatically generates the lot number for the inventory item.

Update_inv_lot

This procedure validates some the input parameters like lot_number, expiration_date, INV attributes, C_ATTRIBUTES, N_ATTRIBUTES and D_ATTRIBUTES. If all the validations go through fine, data is updated with the attribute information.

Validate_unique_lot

This function validates a given lot number for an organization and item depending on the uniqueness level set and returns true or false.

Setting Up the Lot API

Create_Inv_lot

The following chart lists all parameters used by the create_inv_lot procedure. Additional information on these parameters follows the chart.

Parameter	Usage	Type	Required	Derived	Optional

x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_row_id	OUT	RowId			
x_lot_rec	OUT	MTL_Lot_Numbers %RowType			
p_lot_rec	IN	MTL_Lot_Numbers %RowType	x		
p_source	IN	Number	x		
p_api_version	IN	Number	x		x
p_init_msg_list	IN	Varchar2			
p_commit	IN	Varchar2			x
p_validation_level	IN	Number			x
p_origin_trx_id	IN	Number	x		

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Return message count from the error stack in case of failure.

x_msg_data

Return error message in case of failure.

x_row_id

Returns the row id of the Lot Number record in the table.

x_lot_rec

Returns the Lot Number record inserted

p_lot_rec

Indicates a record of type same as MTL_LOT_NUMBERS table and the values in the record will be used to insert rows.

p_source

Indicates which application calls this procedure, whether is OSFM form, OSFM Open Interface or INV.

p_api_version

Indicates the version of the API.

p_init_msg_list

- Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.
- Default Value: FND_API.G_FALSE

p_commit

Indicates whether to commit the changes to the database.

p_validation_level

Validation Level is passed as input in this variable

p_origin_trx_id

Indicates the transaction identifier.

CREATE_INV_LOT

The following chart lists all parameters used by the create_inv_lot procedure. Additional information on these parameters follows the chart.

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_inventory_item_id	IN	Number	x		
p_organization_id	IN	Number	x		
p_lot_number	IN	Varchar2	x		
p_expiration_date	IN	Date		x	
p_disable_flag	IN	Number	x		
p_attribute_category	IN	Varchar2	x		
p_lot_attribute_category	IN	Varchar2	x		
p_attributes_tbl	IN	Char_tbl		x	
p_c_attributes_tbl	IN	Char_tbl		x	

p_n_attributes_tbl	IN	Number_tbl		x
p_d_attributes_tbl	IN	Date_tbl		x
p_grade_code	IN	Varchar2	x	
p_origination_date	IN	Date	x	
p_date_code	IN	Varchar2	x	
p_status_id	IN	Number	x	
p_change_date	IN	Date	x	
p_age	IN	Number	x	
p_retest_date	IN	Date	x	
p_maturity_date	IN	Date	x	
p_item_size	IN	Number	x	
p_color	IN	Varchar2	x	
p_volume	IN	Number	x	
p_volume_uom	IN	Varchar2	x	
p_place_of_origin	IN	Varchar2	x	
p_best_by_date	IN	Date	x	

p_length	IN	Number	x
p_length_uom	IN	Varchar2	x
p_recycled_content	IN	Number	x
p_thickness	IN	Number	x
p_thickness_uom	IN	Varchar2	x
p_width	IN	Number	x
p_width_uom	IN	Varchar2	x
p_territory_code	IN	Varchar2	x
p_supplier_lot_number	IN	Varchar2	x
p_vendor_name	IN	Varchar2	x
p_source	IN	Number	x

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Return message count from the error stack in case of failure

x_msg_data

Return the error message in case of failur.e

p_inventory_item_id

Indicates the inventory item identifier.

p_organization_id

Indicates the organization identifier.

p_lot_number

Lot number to be generated is passed as input in this variable.

p_expiration_date

The Lot expiration date is passed as input in this variable.

p_disable_flag

The disable flag is passed as input in this variable.

p_attribute_category

The descriptive flexfield context is passed as input in this variable.

p_lot_ attribute_category

The Lot Attributes context is passed as input in this variable.

p_attributes_tbl

The Descriptive attributes are passed as input in this variable.

p_c_attributes_tbl

The Character Lot Attributes are passed as input in this variable.

p_n_attributes_tbl

The Numeric Lot Attributes are passed as input in this variable.

p_d_attributes_tbl

The Date Lot Attributes are passed as input in this variable

p_grade_code

The Grade Code Attribute is passed as input in this variable.

p_origination_date

The origination date is passed as input in this variable.

p_date_code

The date code attribute is passed as input in this variable.

p_status_id

The lot status ID is passed as input in this variable.

p_change_date

The Change Date is passed as input in this variable.

P_age

Age attribute is passed as input in this variable.

P_retest_date

Retest Date Attribute is passed as input in this variable.

P_maturity_date

Maturity Date attribute is passed as input in this variable.

P_item_size

Item Size Attribute is passed as input in this variable.

P_color

Color Attribute is passed as input in this variable.

P_volume

Volume attribute is passed as input in this variable.

P_volume_uom

Volume Unit of Measure attribute is passed as input in this variable.

P_place_of_origin

Place of Origin Attribute is passed as input in this variable.

Place of Origin Attribute is passed as input in this variable

Best by Date attribute is passed as input in this variable.

P_length

Length attribute is passed as input in this variable.

P_length_uom

Length Unit of Measure is passed as input in this variable.

P_recycled_content

Recycled Content attribute is passed as input in this variable.

P_thickness

Thickness attribute is passed as input in this variable.

P_thickness_uom

Thickness Unit of Measure attribute is passed as input in this variable.

P_width

Width attribute is passed as input in this variable.

P_width_uom

Width Unit of Measure attribute is passed as input in this variable.

P_territory_code

Territory code attribute is passed as input in this variable.

P_supplier_lot_number

Supplier Lot Number is passed as input in this variable.

P_vendor_name

Vendor Name is passed as input in this variable.

P_source

Indicates which application calls this procedure, whether is OSFM form, OSFM Open Interface or INV.

AUTO_GEN_LOT

The following chart lists all parameters used by the auto_gen_lot procedure. Additional information on these parameters follows the chart.

Parameter	Usage	Type	Required	Derived	Optional
p_org_id	IN	Number	x		
p_inventory_item_id	IN	Number	x		
p_lot_generation	IN	Number			x
p_lot_uniqueness	IN	Number			x
p_lot_prefix	IN	Varchar2			x
p_zero_pad	IN	Number			x
p_lot_length	IN	Number			x
p_transaction_date	IN	Date			x
p_revision	IN	Varchar2			x
p_subinventory_code	IN	Varchar2			x
p_locator_id	IN	Number			x
p_transaction_type_id	IN	Number			x
p_transaction_action_id	IN	Number			x

p_transaction_source_type_id	IN	Number			x
p_lot_number	IN	Varchar2			x
p_api_version	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_commit	IN	Varchar2		x	
p_validation_level	IN	Number		x	
p_parent_lot_number	IN	Varchar2	x		
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Varchar2			
x_msg_data	OUT	Varchar2			

p_org_id

Organization ID is passed as input in this variable.

p_inventory_id

Inventory Item ID is passed as input in this variable.

p_lot_generation

The lot generation flag in the organization is passed in this variable. Possible values are 1 - At organization level, 3 -User defined , 2 - At item level.

p_lot_uniqueness

Lot Uniqueness Level is passed as input in this variable. Possible values are 1-Unique for Item, 2- No Uniquesness Control.

p_lot_prefix

Lot Prefix is passed as input in this variable.

p_zero_pad

Indicates whether to zero padd the lot number name.

P_lot_length

The length of the lot number name is passed as input in this variable.

p_transaction_date

Transaction Date is passed as input in this variable.

p_revision

Revision of the item is passed as input in this variable.

p_subinventory_code

Subinventory Code where the lot number will reside is passed as input in this variable.

p_locator_id

Locator Id is passed as input in this variable

p_transaction_type_id

Transaction Type Id is passed as input in this variable. The possible values can be fetched from MTL_TRANSACTION_TYPES table . The p_transaction_action_id and p_transaction_source_type_id should be populated based on the TRANSACTION TYPE.

p_transaction_action_id

Transaction Action Id is passed as input in this variable.

p_transaction_source_type_id

Transaction Source Type is passed as input in this variable.

p_lot_number

Lot number to be generated is passed as input in this variable.

p_api_version

Indicates the version of the API.

p_init_msg_list

- Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using FND_MSG_PUB.GET.
- Default Value: FND_API.G_FALSE

p_commit

Indicates whether to commit the changes to the database.

p_validation_level

Validation Level is passed as input in this variable.

p_parent_lot_number

Parent Lot Number is passed as input in this variable.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Return message count from the error stack in case of failure.

x_msg_data

Return the error message in case of failure.

UPDATE_INV_LOT

The following chart lists all parameters used by the update_inv_lot procedure. Additional information on these parameters follows the chart.

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	Varchar2	OUT			
x_msg_count	Number	OUT			
x_msg_data	Varchar2	OUT			
p_inventory_item_id	Number	IN	x		
p_organization_id	Number	IN	x		
p_lot_number	Varchar2	IN	x		
p_expiration_date	Date	IN		x	
p_disable_flag	Number	IN	x		
p_attribute_category	Varchar2	IN	x		
p_lot_attribute_category	Varchar2	IN	x		
p_attributes_tbl	Char_tbl	IN		x	
p_c_attributes_tbl	Char_tbl	IN		x	
p_n_attributes_tbl	Number_tbl	IN		x	
p_d_attributes_tbl	Date_tbl	IN		x	
p_grade_code	Varchar2	IN	x		

p_origination_date	Date	IN	x
p_date_code	Varchar2	IN	x
p_status_id	Number	IN	x
p_change_date	Date	IN	x
p_age	Number	IN	x
p_retest_date	Date	IN	x
p_maturity_date	Date	IN	x
p_item_size	Number	IN	x
p_color	Varchar2	IN	x
p_volume	Number	IN	x
p_volume_uom	Varchar2	IN	x
p_place_of_origin	Varchar2	IN	x
p_best_by_date	Date	IN	x
p_length	Number	IN	x
p_length_uom	Varchar2	IN	x
p_recycled_content	Number	IN	x
p_thickness	Number	IN	x

p_thickness_uom	Varchar2	IN	x
p_width	Number	IN	x
p_width_uom	Varchar2	IN	x
p_territory_code	Varchar2	IN	x
p_supplier_lot_number	Varchar2	IN	x
p_vendor_name	Varchar2	IN	x
p_source	Number	IN	x

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Return message count from the error stack in case of failure.

x_msg_data

Return the error message in case of failure.

p_inventory_item_id

Indicates the inventory item identifier.

p_organization_id

Indicates the organization identifier.

p_lot_number

Lot number to be generated is passed as input in this variable.

p_expiration_date

The Lot expiration date is passed as input in this variable.

p_disable_flag

The disable flag is passed as input in this variable. Possible values are 1- Yes, 2 -No.

p_attribute_category

The descriptive flexfield context is passed as input in this variable.

p_lot_ attribute_category

The Lot Attributes context is passed as input in this variable.

p_attributes_tbl

The Descriptive attributes are passed as input in this variable.

p_c_attributes_tbl

The Character Lot Attributes are passed as input in this variable.

p_n_attributes_tbl

The Numeric Lot Attributes are passed as input in this variable.

p_d_attributes_tbl

The Date Lot Attributes are passed as input in this variable.

p_grade_code

The Grade Code Attribute is passed as input in this variable.

p_origination_date

The origination date is passed as input in this variable.

p_date_code

The date code attribute is passed as input in this variable.

p_status_id

The lot status ID is passed as input in this variable.

p_change_date

The Change Date is passed as input in this variable.

P_age

Age attribute is passed as input in this variable.

P_retest_date

Retest Date Attribute is passed as input in this variable.

P_maturity_date

Maturity Date attribute is passed as input in this variable.

P_item_size

Item Size Attribute is passed as input in this variable.

P_color

Color Attribute is passed as input in this variable.

P_volume

Volume attribute is passed as input in this variable.

P_volume_uom

Volume Unit of Measure attribute is passed as input in this variable.

P_place_of_origin

Place of Origin Attribute is passed as input in this variable.

P_best_by_date

Best by Date attribute is passed as input in this variable.

P_length

Length attribute is passed as input in this variable.

P_length_uom

Length Unit of Measure is passed as input in this variable.

P_recycled_content

Recycled Content attribute is passed as input in this variable.

P_thickness

Thickness attribute is passed as input in this variable.

P_thickness_uom

Thickness Unit of Measure attribute is passed as input in this variable.

P_width

Width attribute is passed as input in this variable.

P_width_uom

Width Unit of Measure attribute is passed as input in this variable.

P_territory_code

Territory code attribute is passed as input in this variable.

P_supplier_lot_number

Supplier Lot Number is passed as input in this variable.

P_vendor_name

Vendor Name is passed as input in this variable.

P_source

Indicates which application calls this procedure, whether is OSFM form, OSFM.

VALIDATE_UNIQUE_LOT

The following chart lists all parameters used by the validate_unique_lot function. Additional information on these parameters follows the chart.

Parameter	Usage	Type	Required	Derived	Optional

p_org_id	IN	Number	x
p_inventory_item_id	IN	Number	x
p_lot_uniqueness	IN	Varchar2	x
p_auto_lot_number	IN	Varchar2	x

p_org_id

Organization ID is passed as input in this variable.

p_inventory_item_id

Inventory Item ID is passed as input in this variable.

p_lot_uniqueness

Lot Uniqueness Level is passed as input in this variable. Possible values are 1- Unique for Item, 2- No Uniqueness Control.

p_auto_lot_number

Lot number to be validated is passed as input in this variable.

Validation of Lot API

Standard Validation

Oracle Inventory validates all input parameters in the Lot API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return error status to the calling module. The Pick Release API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Material Reservation Application Program Interface

The Material Reservation API is a public API that allows you to do the following:

- Query existing reservations
- Create reservations
- Update existing reservations
- Transfer existing reservations
- Delete existing reservations

The Material Reservation API provides the following public procedures that allow you to accomplish the previous tasks:

- query_reservation

  Returns all reservations that match the specified criteria.

- query_reservation_om_hdr_line

  This API is exclusively for Order Management, who query reservations extensively.

- create_reservation

  Creates a material reservation for an item.

- update_reservation

  Updates the demand and supply information, including quantity, of an existing reservation.

- transfer_reservation

  Transfers either supply or demand information from one source to another.

- relieve_reservation

  This function relieves an existing reservation based on the parameters passed to the API.

- delete_reservation

  Deletes existing reservations. Used when a reservation is no longer needed or has been fulfilled.

Setting Up the Material Reservation API

The following charts list all parameters used by the procedures listed above. Additional information on these parameters follows each chart.

QUERY_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_query_input	IN	Record	x		
p_lock_records	IN	Varchar2	x		
p_sort_by_rec_date	IN	Number	x		

Parameter	Usage	Type	Required	Derived	Optional
p_cancel_order_mode	IN	Number	x		
x_mtl_reservation_tbl	OUT	PL/SQL Table			
x_mtl_reservation_tbl_count	OUT	Number			
x_error_code	OUT	Number			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_query_input

Contains information to be used to identify the reservations.

p_lock_records

Specifies whether to lock matching records.

Default Value: FND_API.G_FALSE

p_sort_by_req_date

Specifies whether to sort the return records by requirement date.

Default Value: INV_RESERVATION_GLOBAL.G_QUERY_NO_SORT

p_cancel_order_mode

If you intend to cancel an order and want to query related reservations, the reservations will be returned in a specific order.

Default Value: INV_RESERVATION_GLOBAL.G_CANCEL_ORDER_NO

x_mtl_reservation_tbl

Indicates reservations that match the criteria.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_TBL_TYPE

x_mtl_reservation_tbl_count

Indicates the number of records in x_mtl_reservation_tbl.

x_error_code

This error code is meaningful only if x_return_status equals fnd_api.g_ret_sts_error.

CREATE_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
p_query_input	IN	Record	x		
p_lock_records	IN	Varchar2	x		
p_sort_by_rec_date	IN	Number	x		
p_cancel_order_mode	IN	Number	x		
x_mtl_reservation_tbl	OUT	PL/SQL Table			
x_mtl_reservation_tbl_count	OUT	Number			
x_error_code	OUT	Number			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_rsv_rec

Contains information to create the reservations.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

p_serial_number

Contains serial numbers to be reserved.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

X_serial_number

The serial numbers actually reserved if procedure succeeded.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

p_partial_reservation_flag

If not enough quantity is available, specifies whether to reserve the amount that is available. Possible values are FND_API_.G_FALSE and FND_API_.G_TRUE

Default Value: FND_API_.G_FALSE

p_force_reservation_flag

Specifies whether to reserve the quantity without performing a quantity check.

Default Value: FND_API_.G_FALSE

p_validation_flag

Specifies whether to reserve the quantity without performing a validation.

Default Value: FND_API_.G_TRUE.

x_quantity_reserved

The actual quantity reserved if the procedure succeeded.

x_reservation_id

This reservation identifier for the reservation is created if the procedure succeeded.

P_VALIDATION_FLAG

This flag indicates whether or not to reserve without validation.

Default Value: FND_API_.G_TRUE.

P_OVER_RESERVATION_FLAG

This flag indicates whether to allow over reservation or not.

Default Value: 0

X_SECONDARY_QUANTITY_RESERVED

The quantity actual reserved if succeeded in secondary UOM.

P_PARTIAL_RSV_EXISTS

Suggests whether there are existing reservations.

Default Value: FALSE

UPDATE_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_original_rsv_rec	IN	Record	x		
p_to_rsv_rec	IN	Record	x		
p_original_serial_number	IN	PL/SQL Table			

Parameter	Usage	Type	Required	Derived	Optional
p_to_serial_number	IN	PL/SQL Table			
p_validation_flag	IN	Varchar2			
P_CHECK_AVAILABILITY	IN	Varchar2			x
P_OVER_RESERVATION_FLAG	IN	Number			x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_original_rsv_rec

Contains information to identify the existing reservation. If the reservation identifier is passed (not null and not equal to FND_API.G_MISS_NUM), it identifies the existing reservation and all other attributes in this record are ignored.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

p_to_rsv_rec

Contains new values of the attributes to be updated. If the value of an attribute of the existing reservation requires updating, the new value of the attribute is assigned to the attribute in this record.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

p_original_serial_number

Contains the serial numbers reserved by the existing reservation.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

p_to_serial_number

Contains the new serial numbers to be reserved.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

p_validation_flag

Indicates whether to reserve without validation.

Default Value: FND_API.G_TRUE

TRANSFER_RESERVATION

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_is_transfer_supply

Default Value: FND_API.G_TRUE

p_original_rsv_rec

Contains information to identify the existing reservation. If the reservation identifier is passed (not null and not equal to FND_API.G_MISS_NUM), it identifies the existing reservation and all other attributes in this record are ignored.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

p_to_rsv_rec

Contains new values of the attributes to be updated. If the value of an attribute of the existing reservation requires updating, the new value of the attribute is assigned to the attribute in this record.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

p_original_serial_number

Contains the serial numbers reserved by the existing reservation.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

p_to_serial_number

Contains the new serial numbers to be reserved.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

p_validation_flag

Indicates whether to reserve without validation.

Default Value: FND_API.G_TRUE

P_CHECK_AVAILABILITY

This flag will check for availability before updating the reservation.

Default Value: FND_API.G_FALSE

P_OVER_RESERVATION_FLAG

This flag indicates whether over reservation should be allowed or not.

Default Value: 0

UPDATE_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
X_QUANTITY_RESERVED	OUT	NUMBER			
X_SECONDARY_QUANTITY_RESERVED	OUT	NUMBER			
p_original_rsv_rec	IN	Record	x		

p_to_rsv_rec	IN	Record	x		
p_original_serial_number	IN	PL/SQL Table			
p_to_serial_number	IN	PL/SQL Table			
p_validation_flag	IN	Varchar2			
P_PARTIAL_RESERVATION_FLAG	IN	VARCHAR2			
P_PARTIAL_RESERVATION_FLAG	IN	VARCHAR2			x
P_OVER_RESERVATION_FLAG	IN	NUMBER			x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

X_QUANTITY_RESERVED

Returns the quantity that was actually reserved.

X_SECONDARY_QUANTITY_RESERVED

Returns the quantity that was actually reserved in secondary UOM.

p_original_rsv_rec

Contains information to identify the existing reservation. If the reservation identifier is passed (not null and not equal to FND_API.G_MISS_NUM), it identifies the existing reservation and all other attributes in this record are ignored. This is of type INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE.

p_to_rsv_rec

Contains new values of the attributes to be updated. If the value of an attribute of the existing reservation requires updating, the new value of the attribute is assigned to the attribute in this record. This is of type INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE.

p_original_serial_number

Contains the serial numbers reserved by the existing reservation. This is of type INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE.

p_to_serial_number

Contains the new serial numbers to be reserved. This is of type INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE.

p_validation_flag

Indicates whether to reserve without validation. Default Value: FND_API.G_TRUE.

P_CHECK_AVAILABILITY

This flag will check for availability before updating the reservation.

Default Value: FND_API.G_FALSE

P_OVER_RESERVATION_FLAG

This flag indicates whether over reservation should be allowed or not.

Default Value: 0

TRANSFER_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
P_IS_TRANSFER_SUPPLY	IN	VARCHAR2			x
P_ORIGINAL_RSV_REC	IN	Record	x		
P_TO_RSV_REC	IN	Record	x		
P_ORIGINAL_SERIAL_NUMBER	IN	PL/SQL Table			
P_TO_SERIAL_NUMBER	IN	PL/SQL Table			
P_VALIDATION_FLAG VARCHAR2	IN	VARCHAR2			x
P_OVER_RESERVATION_FLAG	IN	NUMBER			x
X_TO_RESERVATION_ID	OUT	NUMBER			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_is_transfer_supply

Default Value: FND_API.G_TRUE

p_original_rsv_rec

Contains information to identify the existing reservation. If the reservation identifier is passed (not null and not equal to FND_API.G_MISS_NUM), it identifies the existing reservation and all other attributes in this record are ignored. This is of type INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE.

p_to_rsv_rec

Contains new values of the attributes to be updated. If the value of an attribute of the existing reservation requires updating, the new value of the attribute is assigned to the attribute in this record. This is of type INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE.

p_original_serial_number

Contains the serial numbers reserved by the existing reservation. This is of type

INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

p_to_serial_number

Contains the new serial numbers to be reserved. This is of type INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE.

p_validation_flag

Indicates whether to reserve without validation. Default Value: FND_API.G_TRUE

P_OVER_RESERVATION_FLAG

Indicates whether over reservation should be allowed or not.

x_to_reservation_id

Indicates the new reservation identifier.

RELIEVE_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
P_API_VERSION_NUMBER	IN	NUMBER	x		
P_INIT_MSG_LST	IN	VARCHAR2		x	
X_RETURN_STATUS	OUT	VARCHAR2			
X_MSG_COUNT	OUT	NUMBER			
X_MSG_DATA	OUT	VARCHAR2			
P_RSV_REC	IN	RECORD	x		
P_PRIMARY_RELIEVED_QUANTITY	IN	NUMBER			

P_SECONDARY_RELIEVED_QUANTITY	IN	NUMBER	
P_RELIEVE_ALL	IN	VARCHAR2	x
P_ORIGINAL_SERIAL_NUMBER	IN	PL/SQL Table	
P_VALIDATION_FLAG	IN	VARCHAR2	x
X_PRIMARY_RELIEVED_QUANTITY OUT NUMBER			
X_SECONDARY_RELIEVED_QUANTITY	OUT	NUMBER	
X_PRIMARY_REMAIN_QUANTITY	OUT	NUMBER	
X_SECONDARY_REMAIN_QUANTITY	OUT	NUMBER	

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

P_RSV_REC

Contains info for identifying the reservation to be relieved. This is of type INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

P_PRIMARY_RELIEVED_QUANTITY

Quantity to relieve in primary uom codeThis parameter is required if p_relieve_all = fnd_api.g_false.

P_SECONDARY_RELIEVED_QUANTITY

Quantity to relieve in secondary uom code.

P_RELIEVE_ALL

Identifies whether to relieve all the reservations.

Default Value: FND_API.G_TRUE

P_ORIGINAL_SERIAL_NUMBER

Contains serial numbers reserved by the existing reservation and to be relieved. This is of type INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE.

P_VALIDATION_FLAG

Whether or not to relieve without validation.

Default Value: FND_API.G_TRUE

X_PRIMARY_RELIEVED_QUANTITY

Quantity relieved by the api in primary uom.

X_SECONDARY_RELIEVED_QUANTITY

Quantity relieved by the api in secondary uom.

X_PRIMARY_REMAIN_QUANTITY

The remain quantity of the reservation in primary uom.

X_SECONDARY_REMAIN_QUANTITY

The remain quantity of the reservation in secondary uom.

DELETE_RESERVATION

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_rsv_rec	IN	Record	x		
p_serial_number	IN				x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_EXC_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_rsv_rec

Contains information to identify the existing reservation.

Default Value: INV_RESERVATION_GLOBAL.MTL_RESERVATION_REC_TYPE

p_serial_number

Contains serial numbers reserved by the existing reservation.

Default Value: INV_RESERVATION_GLOBAL.SERIAL_NUMBER_TBL_TYPE

Validation of Material Reservation API

Standard Validation

Oracle Inventory validates all input parameters in the Material Reservation API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Material Reservation API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Reservations Manager Application Program Interface

The Reservations Manager API is a public API that allows you to call the reservation manager on-line or submit reservation requests concurrently. The Reservations Manager processes reservation requests from the MTL_RESERVATIONS_INTERFACE table for creating, updating, transferring, and deleting reservations. The Reservations Manager API includes the public procedure rsv_interface_manager.

Setting Up the Reservations Manager API

The following chart lists all parameters used by Reservations Manager API. Additional information on these parameters follow the chart.

RSV_INTERFACE_MANAGER

Parameter	Usage	Type	Required	Derived	Optional
x_errbuf	OUT	Varchar2			
x_retcode	OUT	Number			
p_api_version_number	IN	Number	x		
p_init_msg_lst	IN	Varchar2		x	

Parameter	Usage	Type	Required	Derived	Optional
p_form_mode	IN	Varchar2			x

x_errbuf

A mandatory concurrent program parameter.

x_retcode

A mandatory concurrent program parameter.

p_api_version_number

Indicates the API version number.

Default Value: 1

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_form_mode

Specifies whether the Reservations Manager is called from the form. Possible values include:

Y to indicate the Reservations Manager is called from the form.

N to indicate the Reservations Manager is not called from the form

Default Value: N

Validation of Reservations Manager API

Standard Validation

Oracle Inventory validates all input parameters in the Reservations Manager API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Reservations Manager API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Sales Order Application Program Interface

The Sales Order API is a public API that allows you to do the following:

- Create a sales order in Oracle Inventory's local definition of a sales order
- Update a sales order in Oracle Inventory's local definition of a sales order
- Get a corresponding Oracle Order Management order header identifier given a sales order identifier
- Get a corresponding sales order identifier given an Order Management order header identifier.

The Sales Order API provides three public procedures that allow you to accomplish the tasks listed above. The procedures are as follows:

- create_salesorder

Creates a sales order in Oracle Inventory's local definition of a sales order.

- get_oeheader_for_salesorder

  Allows you to get a corresponding Oracle Order Management order header identifier given a sales order identifier. A value of negative one (-1) is returned if the sales order identifierwas created by a system other than Oracle Order Management.

- get_salesorder_for_oeheader

  Allows you to get a corresponding sales order identifier given an Oracle Order Management order header identifier. If there is no matching sales order identifier, a null is returned.

Setting Up the Sales Order API

The following charts list all parameters used by the Sales Order API. Additional information on these parameters follows each chart.

CREATE_SALESORDER

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2	x		
p_segment1	IN	Number	x		
p_segment2	IN	Varchar2	x		
p_segment3	IN	Varchar2	x		
p_validate_full	IN	Number	x		
p_validation_date	IN	Date	x		
x_salesorder_id	OUT	Number			

Parameter	Usage	Type	Required	Derived	Optional
x_msg_data	OUT	Varchar2			
x_msg_count	OUT	Number			
x_return_status	OUT	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

p_segment1

p_segment2

Indicates the order type.

p_segment3

Indicates the order source.

p_validate_full

Indicates whether flexfield APIs are used to create sales order flexfields. When set to a value of 1, flexfield APIs are used to create sales order flexfields. When set to a value of

0, the sales order flexfield is created manually.

Default Value: 1

p_validation_date

Indicates the date of creation.

x_salesorder_id

Indicates the returned sales order identifier.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_msg_count

Indicates the number of error messages the API has encountered.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_SUCCESS
- Error: FND_API.G _EXC_ERROR
- Unexpected Error: FND_API.G_EXC_UNEXPECTED_ERROR

GET_OEHEADER_FOR_SALESORDER

Parameter	Usage	Type	Required	Derived	Optional
p_salesorder_id	IN	Number	x		
x_oe_header_id	OUT	Number			
x_return_status	OUT	Varchar2			

p_salesorder_id

Indicates the sales order identifier

x_oe_header_id

Indicates the Oracle Order Management order header identifier.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_SUCCESS
- Error: FND_API.G _EXC_ERROR
- Unexpected Error: FND_API.G_EXC_UNEXPECTED_ERROR

GET_SALESORDER_FOR_OEHEADER

Parameter	Usage	Type	Required	Derived	Optional
p_oe_header_id	IN	Number	x		

p_oe_header_id

Returns a sales order identifier when an Oracle Order Management order header is passed to it.

Validation of Sales Order API

Standard Validation

Oracle Inventory validates all input parameters in the Sales Order API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Sales Order API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Move Order Application Program Interface

The Move Order API is a public API that allows you to do the following:

- Create a move order header
- Create a move order line
- Process a move order (create or update)
- Lock a move order
- Get a move order for a given header or header identifier
- Process a move order line (cancel or update)

The Move Order API provides the following public procedures that allow you to accomplish the tasks listed above:

- create_move_order_header
- create_move_order_lines
- process_move_order
- lock_move_order
- get_move_order
- process_move_order_line

Setting Up the Move Order API

The following charts list all parameters used by the Move Order API. Additional information on the parameters follows each chart.

CREATE_MOVE_ORDER_HEADER

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_return_values	IN	Varchar2		x	
p_commit	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_trohdr_rec	IN	Record	x		
p_trohdr_val_rec	IN	Record	x		
x_trohdr_rec	OUT	Record			
x_trohdr_val_rec	OUT	Record			
p_validation_flag	IN	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API sends back the values on your behalf.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_trohdr_rec

The record that contains the information to be used to create the move order header.

Default Value: G_MISS_TROHDR_REC

p_trohdr_val_rec

Contains information values rather than internal identifiers used to create the move order header.

Default Value: G_MISS_TRODHDR_VAL_REC

x_trohdr_rec

The information of the move order header that was created.

x_trohdr_val_rec

The information values of the move order header that was created.

p_validation_flag

Flag that indicates whether the data needs to be validated. If this is set to 'N', the move order lines are directly created from the input header record type passed. If this is set to 'Y', the move order header is created after performing the appropriate validations.

CREATE_MOVE_ORDER_LINES

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_return_values	IN	Varchar2		x	
p_commit	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
p_trolin_tbl	IN	PL/SQL Table	x		
p_trolin_val_tbl	IN	PL/SQL Table	x		
x_trolin_tbl	OUT	PL/SQL Table			
x_trolin_val_tbl	OUT	PL/SQL Table			
p_validation_flag	IN	Varchar2			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API sends back the values on your behalf.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_trolin_tbl

A table of records that contains the information to be used to create the move order lines.

Default Value: G_MISS_TROLIN_TBL

p_trolin_val_tbl

Contains information values rather than internal identifiers used to create the move order header.

Default Value: G_MISS_TROLIN_VAL_TBL

x_trolin_tbl

The information of the move order lines that were created.

x_trolin_val_tbl

The information values of the move order lines that were created.

p_validation_flag

Flag that indicates whether the data needs to be validated. If this is set to 'N', the move order lines are directly created from the input header record type passed. If this is set to 'Y', the move order header is created after performing the appropriate validations.

PROCESS_MOVE_ORDER

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_return_values	IN	Varchar2		x	
p_commit	IN	Varchar2			
x_return_status	OUT	Varchar2		x	
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_trohdr_rec	IN	Record	x		
p_trohdr_val_rec	IN	Record	x		
p_trolin_tbl	IN	PL/SQL Table	x		
p_trolin_val_tbl	IN	PL/SQL Table	x		
x_trohdr_rec	OU	Record			
x_trohdr_val_rec	OUT	Record			
x_trolin_tbl	OUT	PL/SQL Table			
x_trolin_val_tbl	OUT	PL/SQL Table			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API sends back the values on your behalf.

Default Value: FND_API.G_FALSE

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_trohdr_rec

The record that contains the information to be used to process the move order header.

Default Value: G_MISS_TROHDR_REC

p_trohdr_val_rec

Contains information values rather than internal identifiers used to process the move order header.

Default Value: G_MISS_TROHDR_VAL_REC

p_trolin_tbl

A table of records that contains the information to process the move order lines.

Default Value: G_MISS_TROLIN_TBL

p_trolin_val_tbl

Contains information values rather than internal identifiers used to process the move order header.

Default Value: G_MISS_TROLIN_VAL_TBL

x_trohdr_rec

The information of the move order header that was created.

x_trohdr_val_rec

The information values of the move order header that was created.

x_trolin_tbl

The information of the move order lines that were created.

x_trolin_val_tbl

The information values of the move order lines that were created.

LOCK_MOVE_ORDER

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_return_values	IN	Varchar2		x	

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_trohdr_rec	IN	Record	x		
p_trohdr_val_rec	IN	Record	x		
p_trolin_tbl	IN	PL/SQL Table	x		
p_trolin_val_tbl	IN	PL/SQL Table	x		
x_trohdr_rec	OUT	Record			
x_trohdr_val_rec	OUT	Record			
x_trolin_tbl	OUT	PL/SQL Table			
x_trolin_val_tbl	OUT	PL/SQL Table			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message

- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API sends back the values on your behalf.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_trohdr_rec

The record that contains the information to lock the move order header.

Default Value: G_MISS_TROHDR_REC

p_trohdr_val_rec

Contains information values rather than internal identifiers used to lock the move order header.

Default Value: G_MISS_TROHDR_VAL_REC

p_trolin_tbl

A table of records that contains the information to create the move order lines.

Default Value: G_MISS_TROLIN_TBL

p_trolin_val_tbl

Contains information values rather than internal identifiers used to lock the move order

header.

Default Value: G_MISS_TROLIN_VAL_TBL

x_trohdr_rec

The information of the move order header that was created.

x_trohdr_val_rec

The information values of the move order header that was created.

x_trolin_tbl

The information of the move order lines that were created.

x_trolin_val_tbl

The information values of the move order lines that were created.

GET_MOVE_ORDER

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_return_values	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_header_id	IN	Number			x (you must indicate either this parameter or the following)

Parameter	Usage	Type	Required	Derived	Optional
p_header	IN	Varchar2			x (you must indicate either this parameter or the preceding)
x_trohdr_rec	OUT	Record			
x_trohdr_val_rec	OUT	Record			
x_trolin_tbl	OUT	PL/SQL Table			
x_trolin_val_tbl	OUT	PL/SQL Table			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API sends back the values on your behalf.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_header_id

The header identifier of the move order that you want to get.

Default Value: FND_API.G_MISS_NUM

p_header

The header description of the move order that you want to get.

Default Value: FND_API.G_MISS_CHAR

x_trohdr_rec

The information of the move order header that was retrieved by the API.

x_trohdr_val_rec

The information values of the move order header that was created.

x_trolin_tbl

The information of the move order lines that were retrieved by the API.

x_trolin_val_tbl

The information values of the move order lines that were retrieved by the API.

PROCESS_MOVE_ORDER_LINE

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_return_values	IN	Varchar2		x	
p_commit	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_trolin_tbl	IN	PL/SQL Table	x		
p_trolin_old_tbl	IN	PL/SQL Table	x		
x_trolin_tbl	OUT	PL/SQL Table	x		

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

where 1_message and 1_msg_index_out are local variables of types Varchar2(2000) and

Number respectively.

Default Value: FND_API.G_FALSE

p_return_values

Requests that the API sends back the values on your behalf.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_TRUE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_trolin_tbl

Contains information to be used to process move order lines.

Validation of Move Order API

Standard Validation

Oracle Inventory validates all input parameters in the Move Order API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Move Order API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Move Order Admin API

The Move Order Admin API (INV_MO_Admin_Pub) is a public API that allows you to do the following:

- Cancel a move order
- Cancel a move order line
- Close a move order
- Close a move order line

The Move Order Admin API provides the following public procedures that allow you to accomplish the tasks listed above:

- cancel_order
- cancel_line
- close_order
- close_line

CANCEL_ORDER

Parameter	Usage	Type	Required	Derived	Optional

p_api_version_number	IN	Number	x	
p_init_msg_list	IN	Varchar2		x
p_commit	IN	Varchar2		x
p_validation_level	IN	Varchar2		x
p_header_id	IN	Varchar2	x	
x_msg_count	OUT	Number	x	
x_msg_data	OUT	Varchar2	x	
x_return_status	OUT	Varchar2	x	

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function. Default Value: FND_API.G_FALSE

p_validation_level

Flag that indicates whether the data needs to be validated.

Default Value: FND_API.G_VALID_FULL

p_header_id

Header_Id column corresponding to move order to be canceled.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

CANCEL_LINE

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	x
p_commit	IN	Varchar2		x	x
p_validation_level	IN	Varchar2		x	
p_line_id	IN	Varchar2	x		
x_msg_count	OUT	Number	x		

x_msg_data	OUT	Varchar2	x
x_return_status	OUT	Varchar2	x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

p_validation_level

Flag that indicates whether the data needs to be validated.

Default Value: FND_API.G_VALID_FULL

p_line_id

Indicates Line_Id column of the move order line to be canceled.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function.

Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

CLOSE_ORDER

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	x
p_commit	IN	Varchar2		x	x
p_validation_level	IN	Varchar2		x	
p_header_id	IN	Varchar2	x		
x_msg_count	OUT	Varchar2	x		
x_msg_data	OUT	Varchar2	x		
x_return_status	OUT	Varchar2	x		

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

p_validation_level

Flag that indicates whether the data needs to be validated.

Default Value: FND_API.G_VALID_FULL

p_header_id

Header_Id column corresponding to move order to be closed.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

CLOSE_LINE

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	x

p_commit	IN	Varchar2		x	x
p_validation_level	IN	Varchar2		x	
p_line_id	OUT	Varchar2	x		
x_msg_count	OUT	Number	x		
x_msg_data	OUT	Varchar2	x		
x_return_status	OUT	Varchar2	x		

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initializes the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function.

Default Value: FND_API.G_FALSE

p_validation_level

Flag that indicates whether the data needs to be validated. Default Value: FND_API.G_VALID_FULL

p_line_id

Indicates Line_Id column of the move order line to be closed.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Line_Details_pub

The Line Details public procedure releases a single move order line. The API takes in a move order line and outputs the pick from and putaway to details for that line. These details are returned in the out parameters in the procedure.

The Line Details (INV_Replenish_Details.Line_Details_Pub) public_procedure releases a single move order line. The API takes in a move order line and outputs the pick from and putaway to details for that line. These details are returned in the output parameters in the procedure.

Name	Usage	Default	Type	Required	Derived	Optional
x_retrun_status	out		VARCHAR2			
x_msg_count	out		NUMBER			
x_msg_data	out		VARCHAR2			
p_line_id	in		NUMBER	x		

Name	Usage	Default	Type	Required	Derived	Optional
x_number_of_rows	out		NUMBER			
x_detailed_qty	out		NUMBER			
x_detailed_qty2	out		NUMBER			
x_revision	out		VARCHAR2			
x_locator_id	out		NUMBER			
x_transfer_to_location	out		NUMBER			
x_lot_number	out		VARCHAR2			
x_expiration_date	out		DATE			
x_transaction_temp_id	out		NUMBER			
p_transaction_header_id	in		NUMBER	x		
p_transaction_mode	in		NUMBER	x		
p_move_order_type	in		NUMBER	x		
p_serial_flag	in		VRCHAR2	x		

Name	Usage	Default	Type	Required	Derived	Optional
P_plan_tasks	in		BOOLEAN			
p_auto_pick_confirm	in		BOOLEAN	x		
p_commit	in	False	BOOLEAN			

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include: Success: FND_API.G_RET_STS_SUCCESS Error: FND_API.G_RET_STS_ERROR Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Contains the error message text if the x_msg_count is equal to 1

p_line_id

This column identifies the move order line to be released.

x_number_of_rows

The number of allocations (rows in mtl_material_transactions_temp) generated by the process

x_detailed_qty

The amount of quantity detailed for the move order line.

x_detailed_qty2

The amount of quantity detailed for the move order line in the item's secondary unit of measure.

x_revision

The revision of the material allocated.

x_locator_id

The location of the material allocated.

x_transfer_to_location

The system suggested locator where the material should be staged after picking.

x_lot_number

The lot number of the material allocated.

x_expiration_date

Expiration date of the lot.

x_transaction_temp_id

The ID of the allocation (row in mtl_material_transaction_temp). If multiple allocations then the first transaction_temp_id is returned.

p_transaction_header_id

Indicates the transaction header ID of a mtl_material_transaction_temp record.

p_transaction_mode

Indicates the transaction mode. a value of 1 refers to on-line, 2 to concurrent, and 3 to background.

p_move_order_type

The type of move order determines what API is called to actually do the pick release function. The inner API differs depending on whether this is a Pick Wave move order, a WIP move order or other type of move order.

p_serial_flag

Indicates whether the item is serial controlled.

p_plan_tasks

Indicates whether tasks should be created.

p_auto_pick_confirm

Overrides the organization level parameter to indicate whether the pick confirm API is automatically called after the records have been pick released.

p_commit

Whether the information is commited by API after completion

Pick Confirm Application Program Interface

The Pick Confirm (INV_Pick_Wave_Pick_Confirm_PUB) API is a public API that allows you to perform pick confirmations on move order line detail records. To transact the records, the API calls the transaction processor, which then updates the move order line delivered quantity as well as shipping and reservation information.

The Pick Confirm API provides one public procedure, pick_confirm, which transfers move order line detail records from the MTL_MATERIAL_TRANSACTIONS_TEMP table into the MTL_MATERIAL_TRANSACTIONS table.

Setting Up the Pick Confirm API

The following chart lists all parameters used by the Pick_Confirm procedure. Additional information on these parameters follows the chart.

PICK_CONFIRM

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	
p_commit	IN	Varchar2		x	
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Varchar2			
x_msg_data	OUT	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
p_move_order_type	IN	Number	x		
p_transaction_mode	IN	Number	x		
p_trolin_tbl	IN				x (you must indicate either this parameter or the succeeding)
p_mold_tbl	IN				x (you must indicate either this parameter or the preceding)
x_mmtt_tbl	OUT				
x_trolin_tbl	OUT				
p_transaction_date	IN	Date			x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F

- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API updates information for you after it completes its function.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API returns the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

p_move_order_type

Indicates the move order type.

p_transaction_mode

Indicates the transaction mode. A value of 1 refers to on-line, 2 to concurrent, and 3 to background.

p_trolin_tbl

Indicates the PL/SQL table from which the move order line records are chosen.

p_mold_tbl

Indicates the PL/SQL table from which the move order line detail records are chosen.

x_mmtt_tbl

Indicates the return value of the p_mold_tbl parameter.

x_trolin_tbl

Indicates the return value of the p_trolin_tbl parameter.

p_transaction_date

Indicates transaction date to be assigned for the MTL_MATERIAL_TRANSACTIONS record(s) created. You can specify any date within an open accounting period for this parameter. Default value: NULL (Transaction Date in MTL_MATERIAL_TRANSACTIONS will be the current date).

Validation of Pick Confirm API

Standard Validation

Oracle Inventory validates all input parameters in the Pick_Confirm procedure. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual for details.

Error Handling

If any validation fails, the API will return an error status to the calling module. The Pick Confirm API processes the rows and reports the following values for every record.

Condition	Message Returned	Meaning of Message Returned
Success	S	Process succeeded
Failure	E	Expected error
Failure	U	Unexpected error

Related Topics

Oracle Applications Message Reference Manual

Pick Release Interface

PROCEDURE NAME: assign_pick_slip_numbers

This procedure assigns pick slip numbers to move order lines identified by the supplied move order header. It is called after cartonization has completed in Oracle Warehouse Management enabled organizations.

PARAMETERS:

Name	Useage	Default	Type	Required	Derived	Optional
x_return_status	OUT		VARCHAR2			
x_msg_count	OUT		NUMBER			
x_msg_data	OUT		VARCHAR2			
p_move_order_header_id	IN	0	NUMBER	X		
p_ps_mode	IN		VARCHAR2	x		
p_grouping_rule_id	IN		NUMBER	x		
p_allow_partial_pick	IN		VARCHAR2	x		

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Contains the error message text if the x_msg_count is equal to 1

p_move_order_header_id

This column identifies the move order header for which pick slip numbers will be assigned.

p_ps_mode

Pick Slip Print Mode: I = Immediate, E = Deferred.

p_grouping_rule_id

Overrides org-level and Move Order header-level grouping rule for generating pick slip numbers

p_allow_partial_pick

TRUE if the pick release process should continue after a line fails to be detailed completely. FALSE if the process should stop and roll back all changes if a line cannot be fully detailed. Printing pick slips as the lines are detailed is only supported if this parameter is TRUE, since a commit must be done before printing.

Quantity Tree Program Interface

Procedure Name: CLEAR_QUANTITY_CACHE

The CLEAR_QUANTITY_CACHE procedure deletes all quantity trees in memory. Call this procedure when you call rollback. Otherwise, the trees in memory may not be in sync with the data in the corresponding database tables.

Procedure Name: QUERY_QUANTITIES

The QUERY_QUANTITIES procedure is used to find the values for quantity onhand, revervable quantity onhand, quantity reserved, quantity suggested, available to transact (ATT), and available to reserve (ATR). If the tree in the database is being queried in transaction mode, and the transaction is a subinventory transfer, then you need to pass the subinventory code of the destination sub in the p_transfer_subinventory_code parameter. In all other cases, set the P_Transfer_Subinventory_Code parameter to NULL. The table below provides the specifications for this API:

Parameter	Usage	Default	Type	Required	Derived	Optional
p_api_version_number	IN		Number	x		
p_init_msg_lst	IN	False	Varchar2		x	
x_return_status	OUT		Varchar2			
x_msg_count	OUT		Number			
x_msg_data	OUT		Varchar2	x		
p_organization_id	IN		Number	x		
p_inventory_item_id	IN		Number	x		
p_tree_mode	IN		Integer	x		
p_is_revision_control	IN		Boolean	x		
p_is_lot_control	IN		Boolean	x		
p_is_serial_control	IN		Boolean	x		
p_demand_source_type_id	IN	-9999	Number			x
p_demand_source_header_id	IN	-9999	Number			x

Parameter	Usage	Default	Type	Required	Derived	Optional
p_demand_source_line_id	IN	-9999	Number			x
p_demand_source_name	IN	Null	Varchar2			x
p_lot_expiration_date	IN	Null	Date			x
p_revision	IN		Varchar2	x		
p_lot_number	IN		Varchar2	x		
p_subinventory_code	IN		Varchar2	x		
p_locator_id	IN		Number	x		
p_onhand_source	IN	3	Number		x	
x_qoh	OUT		Number			
x_rqoh	OUT		Number			
x_qr	OUT		Number			
x_qs	OUT		Number			
x_att	OUT		Number			
x_atr	OUT		Number			
p_transfer_subinventory_code	IN	Null	Varchar2	Null		x

Parameter	Usage	Default	Type	Required	Derived	Optional
p_cost_group_id	IN	Null	Number			x
p_lpn_id	IN	Null	Number			x
p_transfer_locator_id	IN	Null	Number			x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests API initialization of the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

x_return_status

Returns Fnd_Api.G_Ret_Sts_Success for success, Fnd_Api.G_Ret_Sts_error for failure to process request due to incorrect input or for an unexpected error fnd_Api.G_Ret_Sts_Unexp_Error.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Contains the error message if x_msg_count =1.

p_organization_id

This column identifies the internal identifier of the Organization.

p_inventory_item_id

This column identifies the internal identifier of the Inventory item.

p_tree_mode

Either reservation mode (1), Transaction mode, (2), Loose Only mode (3) or No LPN Reservation mode (4).

p_is_revision_control

Item is Revision Controlled.

p_is_lot_control

Item is Lot Controlled.

p_is_serial_control

Item is Serial Controlled.

p_demand_source_type_id

Demand Source Type ID.

p_demand_source_header_id

Demand Source Header ID.

p_demand_source_line_id

Demand Source Line ID.

p_demand_source_name

Demand Source Name.

p_lot_expiration_date

Only consider lots that will not expire at or before this date.

p_revision

Item Revision.

p_lot_number

Lot Number.

p_subinventory_code

Subinventory Code.

p_locator_id

Locator ID.

p_onhand_source

Describes subinventories in which to search for onhand: ATPable (1), Nettable (2), All (3), ATPable or Nettable (4).

x_qoh

Quantity On Hand.

x_rqoh

Reservable Quantity On Hand.

x_qr

Quantity Reserved.

x_qs

Quantity Suggested.

x_att

Quantity Available to Transact.

p_transfer_subinventory_code

Destination Subinventory for Subinventory Transfer transactions. Should be NULL otherwise.

p_cost_group_id

Cost Group ID.

p_lpn_id

LPN ID.

p_transfer_locator_id

Destination Locator for Subinventory Transfer transactions. Should be NULL otherwise.

Procedure Name: UPDATE_QUANTITIES

The UPDATE_QUANTITIES procedure changes the quantity in the quantity tree at the level specified by the input. It returns the quantities at the level after the update. The quantity updated depends on the quantity type parameter. If the quantity type is g_qoh, then the p_primary_quantity value is added to the quantity onhand. If the quantity type g_qs_txn, then the quantity suggested value is updated. Update_quantity does not update the database. The UPDATE_QUANTITIES procedure updates the local version of the quantity tree. The database must be updated separately.

- The quantity passed into update_quantities is important. The quantity is always added to the appropriate node quantity. For a receipt transaction, the quantity passed in should be positive. For an issue transaction, the quantity passed in should have a negative sign (to decrement on hand quantity). For reserving items or suggesting an issue, the value passed in should be positive incrementing quantity reserved or quantity suggested. Do not update the tree with suggested receipts. Including suggested receipts could lead to missing inventory if the suggestion is not transacted

- For a subinventory transfer transaction, which updates both the destination location and the source location, UPDATE_QUANTITIES must be called twice. First, add the quantity to the destination sub or location. Then decrement the quantity from the source sub or location. The updates to both the destination and source should happen for actual transactions, not suggested transfers. The table below provides the specifications for this API:

Parameter	Usage	Default	Type	Required	Derived	Optional
p_api_version_number	IN		Number	x		
p_init_msg_lst	IN	False	Varchar2		x	
x_return_status	OUT		Varchar2			
x_msg_count	OUT		Number			
x_msg_data	OUT		Varchar2	x		
p_organization_id	IN		Number	x		
p_inventory_item_id	IN		Number	x		
p_tree_mode	IN		Integer	x		
p_is_revision_control	IN		Boolean	x		
p_is_lot_control	IN		Boolean	x		
p_is_serial_control	IN		Boolean	x		
p_demand_source_type_id	IN	-9999	Number			x
p_demand_source_header_id	IN	-9999	Number			x

Parameter	Usage	Default	Type	Required	Derived	Optional
p_demand_source_line_id	IN	-9999	Number			x
p_demand_source_name	IN	Null	Varchar2			x
p_lot_expiration_date	IN	Null	Date			x
p_revision	IN		Varchar2	x		
p_lot_number	IN		Varchar2	x		
p_subinventory_code	IN		Varchar2	x		
p_locator_id	IN		Number	x		
p_grade_code	IN		Varchar2	x		
p_primary_quantity	IN		Number	x		
p_quantity_type	IN		Integer	x		
p_onhand_source	IN	3	Number		x	
x_qoh	OUT		Number			
x_rqoh	OUT		Number			
x_qr	OUT		Number			
x_qs	OUT		Number			

Parameter	Usage	Default	Type	Required	Derived	Optional
x_att	OUT		Number			
x_atr	OUT		Number			
p_transfer_subinventory_code	IN	Null	Varchar2	Null		x
p_cost_group_id	IN	Null	Number			x
p_containerized	IN	Null	Number			
p_lpn_id	IN	Null	Number			x
p_transfer_locator_id	IN	Null	Number			x

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Returns fnd_api.g_ret_sts_success for success, fnd_api.g_ret_sts_error for failure to process request due to incorrect input or for an unexpected error fnd_api.g_ret_sts_unexp_error.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Contains the error message if x_msg_count =1.

p_organization_id

Organization id.

p_inventory_item_id

Inventory item id.

p_tree_mode

Either reservation mode (1), Transaction mode, (2), Loose Only mode (3) or No LPN Reservation mode (4).

p_is_revision_control

Item is Revision Controlled.

p_is_lot_control

Item is Lot Controlled.

p_is_serial_control

Item is Serial Controlled.

p_demand_source_type_id

Demand Source Type ID.

p_demand_source_header_id

Demand Source Header ID.

p_demand_source_line_id

Demand Source Line ID.

p_demand_source_name

Demand Source Name.

p_lot_expiration_date

Only consider lots that will not expire at or before this date.

p_revision

Item Revision.

p_lot_number

Lot Number.

p_subinventory_code

Subinventory Code.

p_locator_id

Locator ID.

p_primary_quantity

Quantity to update tree with (in primary UOM).

p_quantity_type

Used to specify which quantity to change: quantity onhand (1), quantity reserved same demand source (2), quantity reserved other demand (3), quantity suggested for reservation (4), quantity suggested for transaction (5).

p_onhand_source

Describes subinventories in which to search for onhand: ATPable (1), Nettable (2), All (3), ATPable or Nettable (4).

x_qoh

Quantity On Hand.

x_rqoh

Reservable Quantity On Hand.

x_qr

Quantity Reserved.

x_qs

Quantity Suggested.

x_att

Quantity Available to Transact.

p_transfer_subinventory_code

Destination Subinventory for Subinventory Transfer transactions. Should be NULL otherwise.

p_cost_group_id

Cost Group ID.

p_lpn_id

LPN ID.

p_transfer_locator_id

Destination Locator for Subinventory Transfer transactions. Should be NULL otherwise.

Procedure Name: DO_CHECK

The DO_CHECK procedure checks to determine whether tree updates are still valid. It should be called before committing quantity updates to the database. There can be multiple quantity trees for each item and organization. Updates in a quantity tree are not reflected in other quantity trees of the same organization or item. Thus, it would be possible for two different sessions to try to reserve or transact the same quantity. Either session would lead to a negative quantity. To solve this problem, call do-check before ccommitting The DO_CHECK procedure rebuilds the quantity tree with the current information in the database. If your updates result in negative quantities (and if negative quantities are not allowed), then x_no_violation will be false. Updates should then be rolled back. The table below provides the specifications for this procedure:

Parameter	Usage	Default	Type	Required	Derived	Optional
p_api_version_number	IN		Number	x		
p_init_msg_lst	IN	False	Varchar2		x	
x_return_status	OUT		Varchar2			
x_msg_count	OUT		Number			

Parameter	Usage	Default	Type	Required	Derived	Optional
x_msg_data	OUT		Varchar2			
x_no_violation	OUT		Boolean			

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

x_return_status

Returns fnd_api.g_ret_sts_success for success, fnd_api.g_ret_sts_error for failure to process request due to incorrect input or for an unexpected error fnd_api.g_ret_sts_unexp_error.

x_msg_count

Indicates the number of error messages the API has encountered.

x_msg_data

Contains the error message if x_msg_count =1.

x_no_violation

Return true if no validation found, otherwise value equal false.

Procedure Name: QUERY_QUANTITIES

The QUERY_QUANTITIES procedure is used to find the values for quantity onhand, reservable quantity onhand, quantity reserved, quantity suggested, available to transact (ATT), and available to reserve (ATR). If the tree in the database is being queried in transaction mode, and the transaction is a subinventory transfer, then you need to pass the subinventory code of the destination sub in the p_transfer_subinventory_code parameter. In all other cases, set the P_Transfer_Subinventory_Code parameter to NULL. The table below provides the specifications for this API:

Parameter	Usage	Default	Type	Required	Derived	Optional
p_api_version_number	IN		Number	x		
p_init_msg_lst	IN	False	Varchar2		x	
x_return_status	OUT		Varchar2			
x_msg_count	OUT		Number			
x_msg_data	OUT		Varchar2	x		
p_organization_id	IN		Number	x		
p_inventory_item_id	IN		Number	x		
p_tree_mode	IN		Integer	x		
p_is_revision_control	IN		Boolean	x		

p_is_lot_control	IN		Boolean	x		
p_is_serial_control	IN		Boolean	x		
p_grade_code	IN		Varchar2	x		
p_demand_source_type_id	IN	-9999	Number			x
p_demand_source_header_id	IN	-9999	Number			x
p_demand_source_line_id	IN	-9999	Number			x
p_demand_source_name	IN	Null	Varchar2			x
p_lot_expiration_date	IN	Null	Date			x
p_revision	IN	Null	Varchar2			
p_lot_number	IN		Varchar2	x		
p_subinventory_code	IN		Varchar2	x		
p_locator_id	IN		Number	x		
p_onhand_source	IN	3	Number	x		
x_qoh	OUT		Number		x	

x_rqoh	OUT		Number	
x_qr	OUT		Number	
x_qs	OUT		Number	
x_att	OUT		Number	
x_atr	OUT		Number	
x_sqoh	OUT		Number	
x_srqoh	OUT		Number	
x_sqr	OUT		Number	
x_sqs	OUT		Number	
x_satt	OUT		Number	
x_satr	OUT		Number	
p_transfer_subinventory_code	OUT	Null	Varchar2	Null
p_cost_group_id	IN	Null	Number	
p_lpn_id	IN	Null	Number	
p_transfer_locator_id	IN	Null	Number	

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests API initialization of the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

x_return_status

Returns Fnd_Api.G_Ret_Sts_Success for success, Fnd_Api.G_Ret_Sts_error for failure to process request due to incorrect input or for an unexpected error fnd_Api.G_Ret_Sts_Unexp_Error.

x_msg_count

Indicates the number of error messages the API has encountered. x_msg_data Contains the error message if x_msg_count =1.

p_organization_id

This column identifies the internal identifier of the Organization.

p_inventory_item_id

This column identifies the internal identifier of the Inventory item.

p_tree_mode

Either reservation mode (1), Transaction mode, (2), Loose Only mode (3) or No LPN Reservation mode (4)

p_is_revision_control

Item is Revision Controlled.

p_is_lot_control

Item is Lot Controlled.

p_is_serial_control

Item is Serial Controlled.

p_grade

Grade code.

p_demand_source_type_id

Demand Source Type ID.

p_demand_source_header_id .

Demand Source Header ID.

p_demand_source_line_id

Demand Source Line ID.

p_demand_source_name

Demand Source Name.

p_lot_expiration_date

Only consider lots that will not expire at or before this date.

p_revision

Item Revision.

p_lot_number

Lot Number.

p_subinventory_code

Subinventory Code.

p_locator_id

Locator ID.

p_onhand_source

Describes subinventories in which to search for onhand: ATPable (1), Nettable (2), All (3), ATPable or Nettable (4).

x_qoh

Quantity On Hand.

x_rqoh

Reservable Quantity On Hand.

x_qr

Quantity Reserved.

x_qs

Quantity Suggested.

x_att

Quantity Available to Transact.

x_atr

Quantity Available to Reserve.

x_sqoh

Secondary Quantity On Hand.

x_srqoh

Secondary Reservable Quantity On Hand.

x_sqr

Secondary Quantity Reserved.

x_sqs

Secondary Quantity Suggested.

x_satt

Secondary Quantity Available to Transact.

x_satr

Secondary Quantity Available to Reserve.

p_transfer_subinventory_code

Destination Subinventory for Subinventory Transfer transactions. Should be NULL otherwise.

p_cost_group_id

Cost Group ID.

p_lpn_id

LPN ID.

p_transfer_locator_id

Destination Locator for Subinventory Transfer transactions. Should be NULL otherwise.

Validation of Quantity Tree API

Standard Validation

Oracle Inventory validates all input parameters in the Quantity Tree API. For specific information on the data implied by these parameters, see your Oracle Inventory Technical Reference Manual.

Error Handling

If any validation fails, the API will return an error status to the calling module.

Condition	Message Returned	Description
Success	S	Process succeeded
Failure	E	Expected Error
Failure	U	Unexpected Error

Material Transactions Applications Program Interface

The Inventory Transaction Manager Program is a public API, that enables you to process transactions from the interface table, mtl_transactions_interface. It also enables you to process pending transaction from table, Mtl_Material_Transaction_Temp. The API will process the transactions and create a historical record in mtl_material_transactions and update on-hand quantity.

Setting Up Transaction Processing API

Package Name: INV_TXN_MANAGER_PUB

Function Name: PROCESS_TRANSACTION

Return: 0 for sucess, -1 for failure

Description: This function is used to process records in the table Mtl_Transactions_Interface, or Mtl_Material_Transactions_Temp. The assumption is that records being processed via mtl_material_transactions_temp have been validated. Records created through non-Oracle applications should be inserted in Mtl_Transactions_Interface, as they will be validated before being processed.

Parameter	Usage	Default	Type	Required	Derived	Optional
p_api_version_number	IN		Number	x		
p_init_msg_lst	IN	fnd_api.g_false	Varchar2		x	x
p_commit	IN	fnd_api.g_false	Varchar2			x
p_validate_level	IN	fnd_api.g_valid_level_full	Number			x
x_return_status	OUT		Varchar2			
x_msg_count	OUT		Number			
x_msg_data	OUT		Varchar2		x	
x_trans_count	OUT		Number		x	
p_table	IN	1	Number			x
p_header_id	IN		Number	x		

p_api_version_number

Indicates the API version number.

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F

- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2(2000 and Number respectively.

Default Value: FND_API.G_FALSE

p_commit

Requests that the API update information for you after it completes its function. Default Value: FND_API.G_FALSE.

p_validate_level

P_Validate_Level can have two possible values: G_Valid_Level_Full or G_Valid_Level_None. G_valid_level_full, implies the records from Mtl_Transactions_Interface will undergo thorough validations. A value of G_Valid_Level_none is for internal use only. Default value: Fnd_Api.G_Valid_Level_Full.

x_return_status

Returns fnd_api.g_ret_sts_success for success, fnd_api.g_ret_sts_error for failure to process request due to incorrect input or for an unexpected error fnd_api.g_ret_sts_unexp_error.

x_msg_count

Indicates the number of error messages that API has encountered.

x_msg_data

Display error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_trans_count

Number of records processed.

p_table

Value 2 implies the records to be processed reside in mtl_material_transactions_temp. Value 1 implies the records to be processed reside in mtl_transactions_interface. Oracle does not expect third party to invoke this API with a value of 2 (records in mtl_material_transactions_temp).

p_header_id

This is an input parameter corresponding to the transaction_header_id of the batch to be processed from mtl_transactions_interface or mtl_material_transactions_temp.

Inter-Company Transaction Flow Application Program Interface

The Inter-Company Transaction Flow API works with the following entities:

- Intercompany Transaction flows- User can create/update intercompany transactions flows, check whether a Inter-company Transaction Flow exists and can also fetch a valid Inter-company Transaction Flow defined between the provided Start Operating Unit and End Operating Unit. Table: MTL_TRANSACTION_FLOW_HEADERS, MTL_TRANSACTION_FLOW_LINES.

- Intercompany relations- User can update and also validate the intercompany relations information for two operating units. Table: MTL_INTERCOMPANY_PARAMETERS

Intercompany Transaction flow API provides the following public procedures for creating, updating, checking intercompany transaction flows.

- CHECK_TRANSACTION_FLOW- This procedure will return true if an Inter-company Transaction Flow exists between the provided Start Operating Unit and End Operating Unit, which is active on the transaction date provided and of the flow type specified i.e. Shipping or Procuring.

- CREATE_TRANSACTION_FLOW- This procedure is used to create an intercompany transaction flow header, transaction flow lines and intercompany relations together.

- GET_TRANSACTION_FLOW- This API is used to get a valid Inter-company Transaction Flow defined between the provided Start Operating Unit and End Operating Unit, which is active on the transaction date provided and of the flow type specified.

- UPDATE_TRANSACTION_FLOW- This procedure is used to update an intercompany transaction flow.

- UPDATE_TRANSACTION_FLOW_HEADER- This procedure is used to update an intercompany transaction flow header.

- UPDATE_TRANSACTION_FLOW_LINE- This procedure is used to update an intercompany transaction flow line.

The following public procedures are provided to update, validate intercompany relations.

- UPDATE_IC_RELATION- This procedure is used to update the intercompany relation information between two operating units.
- VALIDATE_IC_RELATION_REC- This procedure is used to validate the intercompany relations information for two operating units.

Setting Up the Inter-Company Transaction Flow API

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB.CHECK_TRANSACTION_FLOW. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
P_API_VERSION	IN	NUMBER	x	-	-
P_INIT_MSG_LIST	IN	VARCHAR2	-	x	-
P_START_OPERATING_UNIT	IN	NUMBER	x	-	-
P_END_OPERATING_UNIT	IN	NUMBER	x	-	-
P_FLOW_TYPE	IN	NUMBER	x	-	-
P_ORGANIZATION_ID	IN	NUMBER	x	-	-
P_QUALIFIER_CODE_TBL	IN	NUMBER_TBL	x	-	-
P_QUALIFIER_VALUE_TBL	IN	NUMBER_TBL	x	-	-

P_TRANSACTION_DATE	IN	DATE	x	-	-
X_RETURN_STATUS	OUT	VARCHAR2	x	-	-
X_MSG_COUNT	OUT	NUMBER	x	-	-
X_MSG_DATA	OUT	VARCHAR2	x	-	-
X_HEADER_ID	OUT	NUMBER	x	-	-
X_NEW_ACCOUNTING_FLAG	OUT	VARCHAR2	x	-	-
X_TRANSACTION_FLOW_EXISTS	OUT	VARCHAR2	x	-	-

P_API_VERSION API

Version of this procedure. Current version is 1.0.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_START_OPERATING_UNIT

The start Operating Unit for which the Global Procurement or Shipping flow occurred.

P_END_OPERATING_UNIT

The End Operating Unit for which the Global Procurement or Shipping flow occurred.

P_FLOW_TYPE

To indicate whether this is Global Procurement flow or Shipping flow.

- Shipping Flow
- Procuring Flow

P_ORGANIZATION_ID

Indicates the ship from/ship to organization for Shipping and global procurement flows respectively.

P_QUALIFIER_CODE_TBL

Array of Qualifier Codes.

P_QUALIFIER_VALUE_TBL

Table of Qualifier Value IDs.

P_TRANSACTION_DATE

The date when the transaction will take place.

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_COUNT

Return variable holding the number of error messages returned.

X_MSG_DATA

Return variable holding the error message.

X_HEADER_ID

Return variable holding the header id of the transaction flow found.

X_NEW_ACCOUNTING_FLAG

Return variable indicating whether advanced accounting is being used for the transaction flow.

X_TRANSACTION_FLOW_EXISTS

Return FND_API.G_RET_STS_SUCESS if transaction flow found.

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB. CREATE_TRANSACTION_FLOW. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	X	-	-
X_MSG_DATA	OUT	VARCHAR2	x	-	-
X_MSG_COUNT	OUT	NUMBER	x	-	-
X_HEADER_ID	OUT	NUMBER	x	-	-
X_LINE_NUMBER_TBL	OUT	NUMBER_TBL	x	-	-
P_API_VERSION	IN	NUMBER	x	-	-
P_INIT_MSG_LIST	IN	VARCHAR2	-	x	-
P_VALIDATION_LEVEL	IN	NUMBER	x	-	-
P_START_ORG_ID	IN	NUMBER	x	-	-
P_END_ORG_ID	IN	NUMBER	x	-	-
P_FLOW_TYPE	IN	NUMBER	x	-	-

P_ORGANIZATION_ID	IN	NUMBER	x	-	-
P_QUALIFIER_CODE	IN	NUMBER	x	-	-
P_QUALIFIER_VALUE_ID	IN	NUMBER	x	-	-
P_ASSET_ITEM_PRICING_OPTION	IN	NUMBER	x	-	-
P_EXPENSE_ITEM_PRICING_OPTION	IN	NUMBER	x	-	-
P_NEW_ACCOUNTING_FLAG	IN	VARCHAR2	x	-	-
P_START_DATE	IN	DATE	-	-	x
P_END_DATE	IN	DATE	-	-	x
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	x	-	-
P_ATTRIBUTE1-P_ATTRIBUTE15	IN	VARCHAR2	x	-	-
P_LINE_NUMBER_TBL	IN	NUMBER_TBL	x	-	-

P_FROM_ORG_ID_TBL	IN	NUMBER_TBL	x	-	-
P_FROM_ORGANIZATION_ID_TBL	IN	NUMBER_TBL	x	-	-
P_TO_ORG_ID_TBL	IN	NUMBER_TBL	x	-	-
P_TO_ORGANIZATION_ID_TBL	IN	NUMBER_TBL	x	-	-
P_LINE_ATTRIBUTE_CATEGORY_TBL	IN	VARCHAR2_TBL	x	-	-
P_LINE_ATTRIBUTE1_TBL - P_LINE_ATTRIBUTE15_TBL	IN	VARCHAR2_TBL	x	-	-
P_SHIP_ORGANIZATION_ID_TBL	IN	NUMBER_TBL	x	-	-
P_SELL_ORGANIZATION_ID_TBL	IN	NUMBER_TBL	x	-	-
P_VENDOR_ID_TBL	IN	NUMBER_TBL	x	-	-
P_VENDOR_SITE_ID_TBL	IN	NUMBER_TBL	x	-	-

P_CUSTOMER_ID_TBL	IN	NUMBER_TBL	x	-	-
P_ADDRESS_ID_TBL	IN	NUMBER_TBL	x	-	-
P_CUSTOMER_SITE_ID_TBL	IN	NUMBER_TBL	x	-	-
P_CUST_TRX_TYPE_ID_TBL	IN	NUMBER_TBL	x	-	-
P_IC_ATTRIBUTE_CATEGORY_TBL	IN	VARCHAR2_TBL	x	-	-
P_IC_ATTRIBUTE1_TBL - P_IC_ATTRIBUTE15_TBL	IN	VARCHAR2_TBL	x	-	-
P_REVALUE_AVERAGE_FLAG_TBL	IN	VARCHAR2_TBL	x	-	-
P_FREIGHT_CODE_COMB_ID_TBL	IN	NUMBER_TBL	x	-	-
P_INV_CURRENCY_CODE_TBL	IN	NUMBER_TBL	x	-	-
P_IC_COGS_ACCT_ID_TBL	IN	NUMBER_TBL	x	-	-

P_INV_ACCRUAL_ACCT_ID_TBL	IN	NUMBER_TBL	x	-	-
P_EXP_ACCRUAL_ACCT_ID_TBL	IN	NUMBER_TBL	x	-	-

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message.

X_MSG_COUNT

Return variable holding the number of error messages returned.

X_HEADER_ID

Return variable holding the header id of the transaction flow created.

X_LINE_NUMBER_TBL

Return variable table of transaction flow line numbers created.

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_VALIDATION_LEVEL

Indicates the level of validation to be done. Pass 1.

P_START_ORG_ID

Start operating unit .

P_END_ORG_ID

End Operating Unit.

P_FLOW_TYPE

Indicate whether this is used for Global Procurement flow or Shipping flow 1. Shipping Flow 2. Procuring Flow

P_ORGANIZATION_ID

Indicates the ship from/ship to organization for Shipping and global procurement flows respectively.

P_QUALIFIER_CODE

The qualifier code for this release will be "1" representing 'Category'.

P_QUALIFIER_VALUE_ID

The qualifier value. This is the category_id of the item.

P_ASSET_ITEM_PRICING_OPTION

This is applicable for procurement flow for asset items and can have following values. 1 = PO Price i.e. Price will be derived from the list price of purchase order. 2 = Transfer Price .i.e. Price will be derived from price list.

P_EXPENSE_ITEM_PRICING_OPTION

This is applicable for procurement flow for expense items and can have following values. 1 = PO Price i.e Price will be derived from the list price of purchase order. 2 = Transfer Price i.e. Price will be derived from price list

P_NEW_ACCOUNTING_FLAG

Indicates whether advanced accounting to be used. · For Shipping flow value can be either Y or N. · For Procuring flow only value allowed is Y. When value for this parameter is specified as Y, logical transactions will be created that depicts the logical movement of material between the inventory organizations of operating units involved.

P_START_DATE

The date when the Inter-company Transaction Flow becomes active. Default Value: SYSDATE

P_END_DATE

The date when the when Inter-company Transaction Flow become inactive. Default Value: NULL

P_ATTRIBUTE_CATEGORY

Flexfield Attribute for Transaction Flow Header

P_ATTRIBUTE1 - P_ATTRIBUTE15

Flexfield Attribute for Transaction Flow Header

P_LINE_NUMBER_TBL

Table of line numbers

P_FROM_ORG_ID_TBL

Table of from operating unit of the line nodes

P_FROM_ORGANIZATION_ID_TBL

Table of from organization_id of the line nodes.

P_TO_ORG_ID_TBL

Table of to operating unit of the line nodes.

P_TO_ORGANIZATION_ID_TBL

Table of to organization_id of the line nodes.

P_LINE_ATTRIBUTE_CATEGORY_TBL

Table of Flexfield Attribute for Transaction Flow Lines

P_LINE_ATTRIBUTE1_TBL - P_LINE_ATTRIBUTE15_TBL

Table of Flexfield Attribute for Transaction Flow Lines

P_SHIP_ORGANIZATION_ID_TBL

Table of Shipping organization_id for each Transaction Flow Line

P_SELL_ORGANIZATION_ID_TBL

Table of selling organization_id for each Transaction Flow Line

P_VENDOR_ID_TBL

Table of vendor_id for each Transaction Flow Line

P_VENDOR_SITE_ID_TBL

Table of vendor site id for each Transaction Flow Line

P_CUSTOMER_ID_TBL

Table of customer id for each Transaction Flow Line

P_ADDRESS_ID_TBL

Table of address id for each Transaction Flow Line

P_CUSTOMER_SITE_ID_TBL

Table of customer site id for each Transaction Flow Line

P_CUST_TRX_TYPE_ID_TBL

Table of customer transaction type id for each Transaction Flow Line

P_IC_ATTRIBUTE_CATEGORY_TBL

Table of Flexfield Attribute for Inter company relation

P_IC_ATTRIBUTE1_TBL - P_LINE_ATTRIBUTE15_TBL

Table of Flexfield Attribute for Inter company relation

P_REVALUE_AVERAGE_FLAG_TBL

Table of revalue average flags for Inter company relation

P_FREIGHT_CODE_COMB_ID_TBL

Table of Freight Code Combination Ids for Inter company relation

P_INV_CURRENCY_CODE_TBL

Table of currency code for Inter company relation

P_IC_COGS_ACCT_ID_TBL

Table of COGS account ids for Inter company relation

P_INV_ACCRUAL_ACCT_ID_TBL

Table of accrual account Ids for Inter company relation

P_EXP_ACCRUAL_ACCT_ID_TBL

Table of expense accrual account Ids for Inter company relation Bottom of Form

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB.GET_TRANSACTION_FLOW. All of the inbound and outbound parameters are listed. Additional information on these parameters

follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		
X_MSG_DATA	OUT	VARCHAR2	x		
X_MSG_COUNT	OUT	NUMBER	x		
X_TRANSACTION_FLOWS_TBL	OUT	G_TRANSACTION_FLOW_TBL_TYPE	x		
P_API_VERSION	IN	NUMBER	x		
P_INIT_MSG_LIST	IN	VARCHAR2		x	
P_START_OPERATING_UNIT	IN	NUMBER	x		
P_END_OPERATING_UNIT	IN	NUMBER	x		
P_FLOW_TYPE	IN	NUMBER	x		
P_ORGANIZATION_ID	IN	NUMBER	x		
P_QUALIFIER_CODE_TBL	IN	NUMBER_TBL	x		
P_QUALIFIER_VALUE_TBL	IN	NUMBER_TBL	x		
P_TRANSACTION_DATE	IN	DATE	x		

P_GET_DEFAULT_COST_GROUP	IN	VARCHAR2	x

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message

X_MSG_COUNT

Return variable holding the number of error messages returned.

X_TRANSACTION_FLOWS_TBL

Return variable holding the table of records of all the nodes in between the Start Operating Unit and End Operating Unit.

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_START_OPERATING_UNIT

The start Operating Unit for which the Global Procurement or Shipping occurred.

P_END_OPERATING_UNIT

The End Operating Unit for which the Global Procurement of Shipping occurred.

P_FLOW_TYPE

To indicate whether this is Global Procurement flow or Shipping flow.

1. Shipping Flow

2. Procuring Flow

P_ORGANIZATION_ID

Indicates the ship from/ship to organization for shipping and global procurement flows respectively.

P_QUALIFIER_CODE_TBL

Array of Qualifier Codes, The qualifier code for this release will be "1" representing 'Category'.

P_QUALIFIER_VALUE_TBL

Array of Qualifier Value IDs. For this release, it will be the category_id of the item.

P_TRANSACTION_DATE

The date when the transaction will take place.

P_GET_DEFAULT_COST_GROUP

Pass 'Y' to get the default cost group for the intermediate organization nodes.

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB.GET_TRANSACTION_FLOW. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		
X_MSG_DATA	OUT	VARCHAR2	x		
X_MSG_COUNT	OUT	NUMBER	x		
X_TRANSACTION_FLOWS_TBL	OUT	G_TRANSACTION_FLOW_TBL_TYPE	x		

P_API_VERSION	IN	NUMBER	x	
P_INIT_MSG_LIST	IN	VARCHAR2		X
P_HEADER_ID	IN	NUMBER	x	
P_GET_DEFAULT_COST_GROUP	IN	VARCHAR2	x	

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message

X_MSG_COUNT

Return variable holding the number of error messages returned.

X_TRANSACTION_FLOWS_TBL

Return variable holding the table of records of all the nodes in between the Start Operating Unit and End Operating Unit.

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_HEADER_ID

Transaction Flow Header identifier

P_GET_DEFAULT_COST_GROUP

Pass 'Y' to get the default cost group for the intermediate organization nodes.

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB. UPDATE_TRANSACTION_FLOW. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		
X_MSG_DATA	OUT	VARCHAR2	x		
X_MSG_COUNT	OUT	NUMBER	x		
P_API_VERSION	IN	NUMBER	x		
P_INIT_MSG_LIST	IN	VARCHAR2	-	x	
P_VALIDATION_LEVEL	IN	NUMBER	x		
P_HEADER_ID	IN	NUMBER	x		
P_FLOW_TYPE	IN	NUMBER	x		
P_START_DATE	IN	DATE	x		
P_END_DATE	IN	DATE	x		

P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	x
P_ATTRIBUTE1 - P_ATTRIBUTE15	IN	VARCHAR2	x
P_LINE_NUMBER_TBL	IN	NUMBER_TBL	x
P_LINE_ATTRIBUTE_CATEGORY_TBL	IN	VARCHAR2_TBL	x
P_LINE_ATTRIBUTE1_TBL – P_LINE_ATTRIBUTE15_TBL	IN	VARCHAR2_TBL	x
P_SHIP_ORGANIZATION_ID_TBL	IN	NUMBER_TBL	x
P_SELL_ORGANIZATION_ID_TBL	IN	NUMBER_TBL	x
P_VENDOR_ID_TBL	IN	NUMBER_TBL	x
P_VENDOR_SITE_ID_TBL	IN	NUMBER_TBL	x
P_CUSTOMER_ID_TBL	IN	NUMBER_TBL	x
P_ADDRESS_ID_TBL	IN	NUMBER_TBL	x
P_CUSTOMER_SITE_ID_TBL	IN	NUMBER_TBL	x

P_CUST_TRX_TYPE_ID_TBL	IN	NUMBER_TBL	x
P_IC_ATTRIBUTE_CATEGORY_TBL	IN	VARCHAR2_TBL	x
P_IC_ATTRIBUTE1_TBL - P_IC_ATTRIBUTE15_TBL	IN	VARCHAR2_TBL	x
P_REVALUE_AVERAGE_FLAG_TBL	IN	VARCHAR2_TBL	x
P_FREIGHT_CODE_COMB_ID_TBL	IN	NUMBER_TBL	x
P_INV_CURRENCY_CODE_TBL	IN	NUMBER_TBL	x
P_IC_COGS_ACCT_ID_TBL	IN	NUMBER_TBL	x
P_INV_ACCRUAL_ACCT_ID_TBL	IN	NUMBER_TBL	x
P_EXP_ACCRUAL_ACCT_ID_TBL	IN	NUMBER_TBL	x

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message

X_MSG_COUNT

Return variable holding the number of error messages returned

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_VALIDATION_LEVEL

Indicates the level of validation to be done.

P_HEADER_ID

Transaction flow header

P_FLOW_TYPE

Indicate whether this is used for Global Procurement flow or Shipping flow.

1. Shipping Flow
2. Procuring Flow

P_START_DATE

The date when the Inter-company Transaction Flow become active.

P_END_DATE

The date when the when Inter-company Transaction Flow become inactive.

P_ATTRIBUTE_CATEGORY

Flexfield Attribute for Transaction Flow Header.

P_ATTRIBUTE1 - P_ATTRIBUTE15

Flexfield Attribute for Transaction Flow Header

P_LINE_NUMBER_TBL

Table of line numbers

P_LINE_ATTRIBUTE_CATEGORY_TBL

Table of Flexfield Attribute for Transaction Flow Lines

P_LINE_ATTRIBUTE1_TBL - P_LINE_ATTRIBUTE15_TBL

Table of Flexfield Attribute for Transaction Flow Lines

P_SHIP_ORGANIZATION_ID_TBL

Table of Shipping organization_id for each Transaction Flow Line

P_SELL_ORGANIZATION_ID_TBL

Table of selling organization_id for each Transaction Flow Line

P_VENDOR_ID_TBL

Table of vendor_id for each Transaction Flow Line

P_VENDOR_SITE_ID_TBL

Table of vendor site id for each Transaction Flow Line

P_CUSTOMER_ID_TBL

Table of customer id for each Transaction Flow Line

P_ADDRESS_ID_TBL

Table of address id for each Transaction Flow Line

P_CUSTOMER_SITE_ID_TBL

Table of customer site id for each Transaction Flow Line

P_CUST_TRX_TYPE_ID_TBL

Table of customer transaction type id for each Transaction Flow Line

P_IC_ATTRIBUTE_CATEGORY_TBL

Table of Flexfield Attribute for Inter company relation

P_IC_ATTRIBUTE1_TBL - P_IC_ATTRIBUTE15_TBL

Table of Flexfield Attribute for Inter company relation

P_REVALUE_AVERAGE_FLAG_TBL

Table of revalue average flags for Inter company relation

P_FREIGHT_CODE_COMB_ID_TBL

Table of Freight Code Combination Ids for Inter company relation

P_INV_CURRENCY_CODE_TBL

Table of currency code for Inter company relation

P_IC_COGS_ACCT_ID_TBL

Table of COGS account ids for Inter company relation

P_INV_ACCRUAL_ACCT_ID_TBL

Table of accrual account Ids for Inter company relation

P_EXP_ACCRUAL_ACCT_ID_TBL

Table of expense accrual account Ids for Inter company relation

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB.UPDATE_TRANSACTION_FLOW_HEADER. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		
X_MSG_DATA	OUT	VARCHAR2	x		
X_MSG_COUNT	OUT	NUMBER	x		
P_API_VERSION	IN	NUMBER	x		
P_INIT_MSG_LIST	IN	VARCHAR2		x	
P_HEADER_ID	IN	NUMBER	x		
P_END_DATE	IN	DATE	x		

P_START_DATE	IN	DATE	x
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	x
P_ATTRIBUTE1 - P_ATTRIBUTE15	IN	VARCHAR2	x

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message

X_MSG_COUNT

Return variable holding the number of error messages returned

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_HEADER_ID

Transaction flow header

P_END_DATE

The date when the when Inter-company Transaction Flow become inactive.

P_START_DATE

The date when the Inter-company Transaction Flow become active.

P_ATTRIBUTE_CATEGORY

Flexfield Attribute for Transaction Flow Header

P_ATTRIBUTE1 - P_ATTRIBUTE15

Flexfield Attribute for Transaction Flow Header

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB.UPDATE_TRANSACTION_FLOW_LINE. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Paramater	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		
X_MSG_DATA	OUT	VARCHAR2	x		
X_MSG_COUNT	OUT	VARCHAR2	x		
P_API_VERSION	IN	NUMBER	x		
P_INIT_MSG_LIST	IN	VARCHAR2			
P_HEADER_ID	IN	NUMBER	x		
P_LINE_NUMBER	IN	NUMBER	x		
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	x		

P_ATTRIBUTE1 - P_ATTRIBUTE15	IN	VARCHAR2	x

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message

X_MSG_COUNT

Return variable holding the number of error messages returned

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_HEADER_ID

Transaction flow header

P_LINE_NUMBER

Transaction flow line number

P_ATTRIBUTE_CATEGORY

Flexfield Attribute for Transaction Flow Header

P_ATTRIBUTE1 - P_ATTRIBUTE15

Flexfield Attribute for Transaction Flow Header

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB. UPDATE_IC_RELATION. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		
X_MSG_DATA	OUT	VARCHAR2	x		
X_MSG_COUNT	OUT	VARCHAR2	x		
P_API_VERSION	IN	NUMBER	x		
P_INIT_MSG_LIST	IN	VARCHAR2		x	
P_SHIP_ORGANIZATION_ID	IN	NUMBER	x		
P_SELL_ORGANIZATION_ID	IN	NUMBER	x		
P_VENDOR_ID	IN	NUMBER	x		
P_VENDOR_SITE_ID	IN	NUMBER	x		
P_CUSTOMER_ID	IN	NUMBER	x		
P_ADDRESS_ID	IN	NUMBER	x		
P_CUSTOMER_SITE_ID	IN	NUMBER	x		
P_CUST_TRX_TYPE_ID	IN	NUMBER	x		

P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	x
P_ATTRIBUTE1 - P_ATTRIBUTE15	IN	VARCHAR2	x
P_REVALUE_AVERAGE_FLAG	IN	VARCHAR2	x
P_FREIGHT_CODE_COMBINATION_ID	IN	NUMBER	x
P_INV_CURRENCY_CODE	IN	NUMBER	x
P_FLOW_TYPE	IN	NUMBER,	x
P_INTERCOMPANY_COGS_ACCOUNT_ID	IN	NUMBER	x
P_INVENTORY_ACCRUAL_ACCOUNT_ID	IN	NUMBER	x
P_EXPENSE_ACCRUAL_ACCOUNT_ID	IN	NUMBER	x

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message

X_MSG_COUNT

Return variable holding the number of error messages returned

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_SHIP_ORGANIZATION_ID

Shipping organization

P_SELL_ORGANIZATION_ID

Selling organization

P_VENDOR_ID

Vendor id

P_VENDOR_SITE_ID

Vendor site id

P_CUSTOMER_ID

Customer id

P_ADDRESS_ID

Address id

P_CUSTOMER_SITE_ID

Customer site id

P_CUST_TRX_TYPE_ID

This parameter represents the account receivable's transaction type associated with the intercompany relations between two operating units. The transaction type should be the one that is associated with intercompany batch source.

P_ATTRIBUTE_CATEGORY

Flexfield Attribute for Transaction Flow Header

P_ATTRIBUTE1 - P_ATTRIBUTE15

Flexfield Attribute for Transaction Flow Header

P_REVALUE_AVERAGE_FLAG

Revalue average flag for Inter-company relation. This determines whether to revalue the average cost when creating intercompany invoices.

P_FREIGHT_CODE_COMBINATION_ID

Freight Code Combination Id for Inter company relation

P_INV_CURRENCY_CODE

Currency code for Inter-company relation

P_FLOW_TYPE

Indicate whether this is used for Global Procurement flow or Shipping flow.

- Shipping Flow
- Procuring Flow

P_INTERCOMPANY_COGS_ACCOUNT_ID

COGS account id for Inter-company relation.

P_INVENTORY_ACCRUAL_ACCOUNT_ID

Accrual account Id for Inter company relation

P_EXPENSE_ACCRUAL_ACCOUNT_ID

Expense accrual account Id for Inter company relation.

The following table lists all parameters used by the public INV_TRANSACTION_FLOW_PUB. VALIDATE_IC_RELATION_REC. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2	x		

X_MSG_DATA	OUT	VARCHAR2	x	
X_MSG_COUNT	OUT	VARCHAR2	x	
X_VALID	OUT	VARCHAR2	x	
P_API_VERSION	IN	NUMBER	x	
P_INIT_MSG_LIST	IN	VARCHAR2	x	x
P_SHIP_ORGANIZATION_ID	IN	NUMBER	x	
P_SELL_ORGANIZATION_ID	IN	NUMBER	x	
P_VENDOR_ID	IN	NUMBER	x	
P_VENDOR_SITE_ID	IN	NUMBER	x	
P_CUSTOMER_ID	IN	NUMBER	x	
P_ADDRESS_ID	IN	NUMBER	x	
P_CUSTOMER_SITE_ID	IN	NUMBER	x	
P_CUST_TRX_TYPE_ID	IN	NUMBER	x	
P_ATTRIBUTE_CATEGORY	IN	VARCHAR2	x	
P_ATTRIBUTE1 - P_ATTRIBUTE15	IN	VARCHAR2	x	

P_REVALUE_AVERAGE_FLAG	IN	VARCHAR2	x
P_FREIGHT_CODE_COMBINATION_ID	IN	NUMBER	x
P_INV_CURRENCY_CODE	IN	NUMBER	x
P_FLOW_TYPE	IN	NUMBER	x
P_INTERCOMPANY_COGS_ACCOUNT_ID	IN	NUMBER	x
P_INVENTORY_ACCRUAL_ACCOUNT_ID	IN	NUMBER	x
P_EXPENSE_ACCRUAL_ACCOUNT_ID	IN	NUMBER	x

X_RETURN_STATUS

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_DATA

Return variable holding the error message.

X_MSG_COUNT

Return variable holding the number of error messages returned.

P_API_VERSION API

Version of this procedure.

P_INIT_MSG_LIST

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then the list of messages must be retrieved using the call FND_MSG_PUB.GET.

Default Value: FND_API.G_FALSE

P_SHIP_ORGANIZATION_ID

Shipping organization

P_SELL_ORGANIZATION_ID

Selling organization

P_VENDOR_ID

Vendor id

P_VENDOR_SITE_ID

Vendor site id

P_CUSTOMER_ID

Customer id

P_ADDRESS_ID

Address id

P_CUSTOMER_SITE_ID

Customer site id

P_CUST_TRX_TYPE_ID

This parameter represents the account receivable's transaction type associated with the intercompany relations between two operating units. The transaction type should be the one that is associated with intercompany batch source.

P_ATTRIBUTE_CATEGORY

Flexfield Attribute for Transaction Flow Header

P_ATTRIBUTE1 - P_ATTRIBUTE15

Flexfield Attribute for Transaction Flow Header

P_REVALUE_AVERAGE_FLAG

Revalue average flag for Inter-company relation. This determines whether to revalue the average cost when creating intercompany invoices.

P_FREIGHT_CODE_COMBINATION_ID

Freight Code Combination Id for Inter company relation

P_INV_CURRENCY_CODE

Value of this parameter is honoured only for Advanced pricing scenario. This parameter can have following values:

- 1 = Currency code of From Operating Unit. This will generate the invoice in the functional currency of From OU.
- 2 = Currency code of To Operating Unit. This will generate the invoice in the functional currency of To OU.
- 3 = Currency code of Order. This will generate the invoice in order currency.

P_FLOW_TYPE

Indicate whether this is used for Global Procurement flow or Shipping flow.

- Shipping Flow
- Procuring Flow

P_INTERCOMPANY_COGS_ACCOUNT_ID

COGS account id for Inter-company relation

P_INVENTORY_ACCRUAL_ACCOUNT_ID

Accrual account Id for Inter company relation.

P_EXPENSE_ACCRUAL_ACCOUNT_I

Expense accrual account Id for Inter company relation.

Validation of Inter-Company Transaction Flow API

Standard Validation

Oracle Inventory validates all required columns in the Inter-Company Transaction Flow API. For specific information on the data referenced by these columns, see:

Oracle Electronic Technical Reference Manual

Creating intercompany transaction flow

While creating the transaction flow, API will first validate the intercompany relation attributes for the operating units and upon successful validation it will insert data into table MTL_INTERCOMPANY_PARAMETERS. Then it will call private API to create

intercompany transaction flow which will insert data into MTL_TRANSACTION_FLOW_HEADERS and MTL_TRANSACTION_FLOW_LINES.

Error Handling

If any validation fails, the API will return the error status to the calling module.

Serial Number Public Application Program Interface

This API provides various functions for working with serial Numbers. The Serial Numbers procedures allow users to create, update and validate serials. In Addition users can get the uniqueness of a serial number, get the difference between two serial numbers, validate and update serial attributes, and create unit (serial) transactions

The INV_SERIAL_NUMBER_PUB consists of the following procedures and functions:

- GENERATE_SERIALS - Generate a Serial Number. Generate a range of serial numbers.
- IS_SERIAL_UNIQUE - Checks whether a serial is unique or not.
- GET_SERIAL_DIFF - Get the quantities of serial between two serial numbers.
- INCREMENT_SER_NUM - Increments the serial number by single unit.
- VALIDATE_UPDATE_SERIAL_ATT - Validates and updates serial attributes for a serial

Setting Up the Serial Number API

The following table lists all parameters used by the public api INV_SERIAL_NUMBER_PUB.GENERATE_SERIALS. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived
X_RETCODE	Out	VARCHAR2	Yes	No
X_ERRBUF	Out	VARCHAR2	Yes	No
P_ORG_ID	In	NUMBER	Yes	No
P_ITEM_ID	In	NUMBER	Yes	No
P_QTY	In	NUMBER	Yes	No

P_SERIAL_CODE	In	VARCHAR2	Yes	No
P_WIP_ID	In	NUMBER	Yes	No
P_REV	In	VARCHAR2	Yes	No
P_LOT	In	VARCHAR2	Yes	No
P_GROUP_MARK_ID	In	NUMBER	No	Yes
P_LINE_MARK_ID	In	NUMBER	No	Yes

X_RETCODE

Return status indicating success or failure.

X_ERRBUF

Returns the error message in case of failure.

P_ORG_ID

Organization Id is passed as input in this variable.

P_ITEM_ID

Inventory Item id passed as input in this variable.

P_QTY

Quantity passed as input in this variable.

P_SERIAL_CODE

Serial contol code is passed as input in this variable.

P_WIP_ID

Wip entity id is passed as input in this variable. Pass null when not relevant.

P_REV

Revision is passed as input in this variable. Pass null when not relevant.

P_LOT

Lot Number is passed as input in this variable. Pass null when not relevant.

P_GROUP_MARK_ID

Group identifier is passed as input in this variable.

Default Value: NULL

P_LINE_MARK_ID

Line identifier is passed as input in this variable Default Value: Null

The following table describes all parameters that are used by the public function INV_SERIAL_NUMBER_PUB .GENERATE_SERIALS. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

The function returns a Number indicating success or failure; 0 on Success, 1 on Error.

Parameter Name	Usage	Type	Required	Derived
P_ORG_ID	In	Number	Yes	No
P_ITEM_ID	In	Number	Yes	No
P_QTY	In	Number	Yes	No
P_WIP_ID	In	Number	Yes	No
P_REV	In	VARCHAR2	Yes	No
P_LOT	In	VARCHAR2	Yes	No
P_GROUP_MARK_ID	In	Number	No	Yes
P_LINE_MARK_ID	In	Number	No	Yes
X_START_SER	Out	VARCHAR2	Yes	No
X_END_SER	Out	VARCHAR2	Yes	No
X_PROC_MSG	Out	VARCHAR2	Yes	No
P_SKIP_SERIAL	In	VARCHAR2	No	Yes

P_ORG_ID

Organization Id is passed as input in this variable.

P_ITEM_ID

Inventory Item id passed as input in this variable.

P_QTY

Quantity passed as input in this variable.

P_WIP_ID

Wip entity id is passed as input in this variable.

P_REV

Revision is passed as input in this variable.

P_LOT

Lot Number is passed as input in this variable.

P_GROUP_MARK_ID

Group identifier is passed as input in this variable.

P_LINE_MARK_ID

Line identifier is passed as input in this variable.

Default Value: Null

X_START_SER

Return Start serial.

X_END_SER

Return End serial.

X_PROC_MSG

Return Message from the Process-Manager.

P_SKIP_SERIAL

Serial number to be excluded is passed as input in this variable.

Default Value: Null

The following table lists all parameters used by the public API, INV_SERIAL_NUMBER_PUB. IS_SERIAL_UNIQUE . All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

The function returns a Number indicating whether the Serial Number is unique or not; returns 0 if the serial is unique, else returns 1.

Parameter Name	Usage	Type	Required	Derived
P_ORG_ID	In	Number	Yes	No
P_ITEM_ID	In	Number	Yes	No
P_SERIAL	In	VARCHAR2	Yes	No
X_PROC_MSG	Out	VARCHAR2	Yes	No

P_ORG_ID

Organization Id is passed as input in this variable.

P_ITEM_ID

Inventory Item id passed as input in this variable.

P_SERIAL

Serial Number is passed as input in this variable.

X_PROC_MSG

Return Message from the Process-Manager.

The following table describes all parameters that are used by the public function INV_SERIAL_NUMBER_PUB.VALIDATE_UPDATE_SERIAL_ATT. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

The function returns a Number indicating success or failure; 0 on Success, 1 on Error.

Parameter Name	Usage	Type	Required	Derived
X_RETURN_STATUS	Out	VARCHAR2	Yes	No
X_MSG_COUNT	Out	Number	Yes	No
X_MSG_DATA	Out	VARCHAR2	Yes	No

X_VALIDATION_STATUS	Out	VARCHAR2	Yes	No
P_SERIAL_NUMBER	In	VARCHAR2	Yes	No
P_ORGANIZATION_ID	In	Number	Yes	No
P_INVENTORY_ITEM_ID	In	Number	Yes	No
P_SERIAL_ATT_TBL	In	INV_LOT_SEL_ATTR.LOT_SEL_ATTRIBUTES_TBL_TYPE	Yes	No
P_VALIDATE_ONLY	In	Boolean	No	Yes

X_RETURN_STATUS

Return status indicating success or failure.

X_MSG_COUNT

Return message count from the error stack in case of failure.

X_MSG_DATA

Return the error message in case of failure.

X_VALIDATION_STATUS

Return the validation status.

P_SERIAL_NUMBER

Serial number is passed as input in this variable.

P_SERIAL_ATT_TBL

Serial Attributes table is passed as input in this variable. It is of type INV_LOT_SEL_ATTR.LOT_SEL_ATTRIBUTES_TBL_TYPE, which is table of record type INV_LOT_SEL_ATTR.lot_sel_attributes_rec_type. Following is the structure of lot_sel_attributes_rec_type:

Parameter	Type	Required	Derived
COLUMN_NAME	VARCHAR2(50)	No	Yes
COLUMN_TYPE	VARCHAR2(50)	No	Yes
COLUMN_VALUE	fnd_descr_flex_col_usage_vl.default_value %TYPE	No	Yes
REQUIRED	VARCHAR2(10)	No	Yes
PROMPT	VARCHAR2(100)	No	Yes
COLUMN_LENGTH	Number	No	Yes

COLUMN_NAME

Name of the attribute column.

Default Value: Null

COLUMN_TYPE

Type of the attribute column.

Default Value: Null

COLUMN_VALUE

Value

Default Value: Null

REQUIRED

Indicates whether required attribute.

Default Value: 'NULL'

PROMPT

Prompt of the attribute column.

Default Value: NULL

COLUMN_LENGTH

Length of the column.

Default Value: Null

P_VALIDATE_ONLY

TRUE is passed as input in this variable if only validation is required.

Default Value: False

The following table lists all parameters used by the public API, INV_SERIAL_NUMBER_PUB.GET_SERIAL_DIFF. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

The function returns a number giving the quantity of units between the two serial numbers provided as input.

Parameter Name	Usage	Type	Required	Derived
P_FM_SERIAL	In	VARCHAR2	Yes	N
P_TO_SERIAL	In	VARCHAR2	Yes	No

P_FM_SERIAL

From Serial Number is passed as input in this variable.

P_TO_SERIAL

To Serial Number is passed as input in this variable.

The following table lists all parameters used by the public API, INV_SERIAL_NUMBER_PUB.INCREMENT_SER_NUM. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

The function returns the next Serial Number with data type of VARCHAR2.

Parameter Name	Usage	Type	Required	Derived
P_CURR_SERIAL	In	VARCHAR2	Yes	No
P_INC_VALUE	In	Number	Yes	No

P_CURR_SERIAL

Current Serial number is passed as input in this variable.

P_INC_VALUE

Increment value is passed as input in this variable.

Validation of Serial Number

Standard Validation

Oracle Inventory validates all required columns in the INV_SERIAL_NUMBER_PUB.GENERATE_SERIALS API. For specific information on the data referenced by these columns, see:

Oracle Electronic Technical Reference Manual

Error Handling

If the procedure fails, the API will return the error status in X_RETCODE variable and the message in X_ERRBUF variable to the calling module.

User Defined Serial Generation Application Program Interface

The User Defined Serial Generation procedure allow users to create Serial Numbers in the system using the logic defined by them, as opposed to the standard Oracle serial number generation logic. The User Defined Serial Generation API consists of the following entities:

- Serial number generation
- This is a stub API and is called while generating a serial number, if the serial generation level is set as 'User Defined' for a particular organization.
- The stub API can be used to auto-generate Serial numbers using a user defined logic. For example, hexadecimal serial numbers, base-n serial numbers.

The Serial generation stub API allows customized serial generation logic to be called when auto generation of serial numbers is needed. The stub API can also be invoked automatically when serial generation concurrent request is submitted as well as when ctrl+G is used to auto-create serial numbers from mobile devices.

The User Defined Serial Generation API provides following procedure for generating serial numbers:

- GENERATE_SERIAL_NUMBER - User needs to code the user defined logic in this stub procedure and return the serial number.

Once the serial number is returned from this procedure, the necessary validations for creating a serial number in the system will be performed by the calling program before actually creating a serial number with the returned value.

Setting Up the User Defined Serial Generation API

The following table lists all parameters used by the public API,

USER_PKG_SERIAL.GENERATE_SERIAL_NUMBER. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

Parameter Name	Usage	Type	Required	Derived	Optional
X_RETURN_STATUS	OUT	VARCHAR2			
X_MSG_DATA	OUT	VARCHAR2			
X_MSG_COUNT	OUT	NUMBER			
X_SERIAL_NUMBER	OUT	VARCHAR2			
P_ORG_ID	IN	NUMBER	x		
P_ITEM_ID	IN	NUMBER	x		

X_RETURN_STATUS

Return status indicating success or failure.

X_MSG_DATA

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

X_MSG_COUNT

Return message count from the error stack in case of failure.

X_SERIAL_NUMBER

Return the serial number to be generated.

P_ORG_ID

Organization Id is passed as input in this variable.

P_ITEM_ID

Inventory Item id passed as input in this variable.

Users also need to perform the following:

1. Setup the serial generation level on the Organization parameters form to 'User Level'.

2. Enter the user logic for generating the serial number in generate_serial_number procedure of the stub api (user_pkg_serial) provided. The procedure needs to return one serial number at a time.

Below is a code sample that generates a serial number based on the Item category set. This example is given as a reference and you should write your own code depending on youir needs. To test the example, you need to create a sequence of name " user_serial_gen_seq_s" in the environment after logging in to sql*plus.

Example

procedure generate_serial_number(x_return_status OUT VARCHAR2, , x_msg_data OUT VARCHAR2 ,x_msg_count OUT NUMBER, x_serial_number OUT VARCHAR2, p_org_id IN NUMBER, p_item_id IN NUMBER)

IS

l_seq NUMBER := NULL; l_prefix varchar2(15) := NULL; BEGIN

Getting the sequence number

SELECT user_serial_gen_seq_s.NEXTVAL INTO l_seq FROM dual;

Getting the prefix

select substr(mcs.category_set_name,1,3) into l_prefix from mtl_category_sets, mtl_item_categories mic where mic.inventory_item_id = p_item_id and mic.organization_id = p_org_id and mcs.category_set_id = mic.cateogory_set_id and rownum < 2;

if l_prefix is null or l_seq is null then raise fnd_api.g_exc_unexpected_error

Setting the return values

x_return_status := fnd_api.g_ret_sts_success; x_serial_number := l_prefix || l_seq; end if;

EXCEPTION when others then

--fnd_message.set_name (...); fnd_msg_pub.ADD; x_return_status := fnd_api.g_ret_sts_error; x_serial_number := NULL; END generate_serial_number;

Validation of User Defined Serial Generation API

Standard Validation

You need to add the following standard validations in the API:

- Serial number created using the User defined logic is returned in X_SERIAL_NUMBER variable.
- The serial Number returned must not be greater than 30 characters.
- Serial Numbers must comprise only characters A-Z, a-z, 0-9.
- [A-Z, a-Z][0-9] is the recommended format for a serial number. It is recommended to have the trailing number as a running decimal number. It is recommended to maintain a database sequence for the actual numeric part. Range transactions are only supported when the serial generation follows the above recommendations.
- Upon successful generation of the serial number using the User defined logic, variable X_RETURN_STATUS returns FND_API.G_RET_STS_SUCCESS. Otherwise it returns FND_API.G_RET_STS_UNEXP_ERROR.

Error Handling

If any validation fails, the API will return the error status to the calling module.

14

Oracle Items Open Interfaces and APIs

Item Open Interface

You can import items from any source into Oracle Inventory and Oracle Engineering using the Item Open Interface. With this interface, you can convert inventory items from another inventory system, migrate assembly and component items from a legacy manufacturing system, convert purchased items from a custom purchasing system, and import new items from a product data management package. The Item Open Interface validates your data, ensuring that your imported items contain the same item detail as items that you enter manually in the Master Item window.

You can also import item category assignments. This can be done simultaneously with a process of importing items, or as a separate task of importing item category assignments only. For this purpose, the Inventory menu contains the Import submenu with the Import Items and Import Item Category Assignments menu entries.

See: Defining Items, *Oracle Inventory User's Guide*.

Functional Overview

The Item Open Interface lets you import items into Oracle Inventory and, if installed at your site, Oracle Engineering. When you import items through the Item Open Interface, you create new items in your item master organization, update existing items, or assign existing items to additional organizations. You can specify values for all the item attributes, or you can specify just a few attributes and let the remainder default or remain null. The Item Open Interface also lets you import revision details, including past and future revisions and effectivity dates. Imported items are validated using the same rules as the item definition windows, so you are ensured of valid items. See: Overview of Engineering Prototype Environment, *Oracle Engineering User's Guide* and Defining Items, *Oracle Inventory User's Guide*.

The Item Open Interface reads data from the following tables for importing items and item details:

- MTL_SYSTEMS_ITEM_INTERFACE: Use this table for your new item numbers and all item attributes. This is the main Item Open Interface table. It may be the only table that you choose to use.
- MTL_ITEM_REVISIONS_INTERFACE: Use this table if you are importing revision details for your new items. This table is used only for revision information. It is not required.
- MTL_ITEM_CATEGORIES_INTERFACE: Use this table to import item category assignments. It stores data about item assignments to category sets, and categories to be imported into the Oracle Inventory MTL_ITEM_CATEGORIES table.
- MTL_DESC_ELEM_VAL_INTERFACE: Use this table to describe elements that apply to your item.

A fifth table, MTL_INTERFACE_ERRORS, is used to record error messages for failed records in the interface table.

Before you use the Item Open Interface, you must write and run a custom program that extracts item information from your source system and inserts the records into the MTL_SYSTEM_ITEM_INTERFACE and table, and (if revision detail is included) the MTL_ITEMS_REVISIONS_INTERFACE and MTL_ITEM_CATEGORIES_INTERFACE tables. After you load item, revision, and item category assignment records into these interface tables, run the Item Open Interface to import the data. The Item Open Interface assigns defaults, validates data that you include, and then imports the new items.

> **Note:** You can process both master and child item records in the same run. The records will import correctly.

You can also use the Item Open Interface to import item material cost, material overhead, and revision details.

Setting Up the Item Open Interface

Create Indexes for Performance

You should create the following indexes to improve Item Open Interface performance.

First, determine which segments are enabled for the System Items flexfield.

Then, for example, if you had a two-segment flexfield, with SEGMENT8 and SEGMENT12 enabled, you would do the following:

```
SQL> create unique index MTL_SYSTEM_ITEMS_B_UC1 on MTL_SYSTEM_ITEMS_B
(ORGANIZATION_ID, SEGMENT8, SEGMENT12);


SQL> create unique index MTL_SYSTEM_ITEMS_INTERFACE_UC1 on
MTL_SYSTEM_ITEMS_INTERFACE (ORGANIZATION_ID, SEGMENT8, SEGMENT12);
```

If you plan to populate the ITEM_NUMBER column in MTL_SYSTEM_ITEMS_INTERFACE instead of the item segment columns, then do not create the MTL_SYSTEM_ITEMS_INTERFACE_UC1 unique index. Instead, create MTL_SYSTEM_ITEMS_INTERFACE_NC1 non-unique index on the same columns.

Create the following indexes for optimum performance:

MTL_SYSTEM_ITEMS_B

- Non-Unique Index on organization_id, segment*n*

  You need at least one indexed, mandatory segment.

MTL_SYSTEM_ITEMS_INTERFACE

- Non Unique Index on inventory_item_id, organization_id
- Non Unique Index on Item_number
- Unique Index on Transaction_id
- Unique Index on organization_id, segment*n*

  Note: If you are populating organization_code instead of organization_id, then re-create this index as non-unique. It should include the segments that were enabled for the System Item Key flexfield. You can use the created default index if you are using segment1.

MTL_ITEM_REVISIONS_INTERFACE

- Non Unique Index on set_process_id
- Non Unique Index on Transaction_id
- Non Unique Index on Organization_id, Inventory_item_id, Revision

MTL_ITEM_CATEGORIES_INTERFACE

- Non Unique Index on inventory_item_id, category_id
- Non Unique Index on set_process_id
- Unique Index on transaction_id

  Note: Populate _id fields whenever possible. Populating inventory_item_id instead of segment (n) for Update mode significantly improves performance. Populating organization_id instead of organization_code for both Create and Update modes

also reduces processing time.

Start the Concurrent Manager

Because you launch and manage the Item Open Interface concurrent program through the concurrent manager, you must ensure that the concurrent manager is running before you can import any items.

Set Profile Option Defaults

Some columns use profile options as default values. You must set these profiles if you want them to default. See: Inventory Profile Options, *Oracle Inventory User's Guide* and Overview of Setting Up, *Oracle Inventory User's Guide*.

Item Open Interface Run-Time Options

To run the Item Open Interface, select Import Items from the Import submenu of the Inventory menu, or select Import Items in the Request Name field in the All Reports window. See: Importing Customer Items, *Oracle Inventory User's Guide*.

When you run the Item Open Interface, you are prompted for report parameters. These are run-time options for the Item Open Interface:

Run-Time Options

Yes: Run the interface for all organization codes in the Item Open Interface table.

No: Run the interface only for the organization that you are currently in. Item Open Interface rows for organizations other than your current organization are ignored.

Validate Items

Yes: Validate all items and the related data that reside in the interface table that have not yet been validated. If items are not validated, then they are not processed into Oracle Inventory.

No: Do not validate items in the interface table. Note that items that are not validated are not processed into Oracle Inventory. You would use this option if you had previously run the Item Open Interface and responded **Yes** for **Validate Items** and **No** for **Process Items,** and now want to process your items.

Process Items

Yes: All qualifying items in the interface table are inserted into Oracle Inventory.

No: Do not insert items into Oracle Inventory. Use this option, along with **Yes** for **Delete Processed Items**, to remove successfully processed rows from the interface table without performing any other processing. You can also use this option, along with **Yes**

for **Validate Items**, if you want to validate items without any processing.

Delete Processed Rows

Yes: Delete successfully processed items from the Item Open Interface tables.

No: Leave all rows in the Item Open Interface tables.

Process Set

Enter a set ID number for the set of records in the interface table that you want to process. The program picks up the records that have that ID in the SET_PROCESS_ID column. If you leave this field blank, then all rows are picked up for processing regardless of the SET_PROCESS_ID column value.

Create or Update Items

1. Create new items.
2. Update existing items. See: Update Existing Items subsequently..
3. Sync. Transforms into Create if the item does not exist. If the item exists, then it transforms to Update.

You can create or update items by running separate concurrent programs of the Import Items program.

Inserting Item Information into the Item Open Interface Table

Item Open Interface Table Description

The Item Open Interface table MTL_SYSTEM_ITEMS_INTERFACE contains *every* column in the Oracle Inventory item master table**,** MTL_SYSTEM_ITEMS. The columns in the Item Open Interface correspond directly to those in the item master table. Except for ITEM_NUMBER or SEGMENT*n* columns, ORGANIZATION_CODE or ORGANIZATION_ID, DESCRIPTION, PROCESS_FLAG, and TRANSACTION_TYPE, all other columns are optional, either because they have defaults that can be derived, or because the corresponding attributes are optional and may be left null.

The item costing columns (those that begin **MATERIAL**_...) and the REVISION column *are* used for importing item costs and revisions and are discussed in a later section of this chapter.

You can also put in details about other interface tables that are not used by the Item Open Interface.

Currently, the interface does not support the MTL_CROSS_REFERENCE_INTERFACE or MTL_SECONDARY_LOCS_INTERFACE.

The MTL_ITEM_CATEGORIES_INTERFACE is used by the Item Open Interface for

both internal processing of default category assignments *and* to retrieve data populated by the user to be imported into the Oracle Inventory MTL_ITEM_CATEGORIES table.

(Partial List of Columns) Column Name	Type	Required	Derived	Optional
ITEM_NUMBER	Varchar2(81)	*conditionally*		
DESCRIPTION	Varchar2(240)	*conditionally*		
MATERIAL_COST	Number			x
MATERIAL_OVERHEAD_RATE	Number			x
MATERIAL_OVERHEAD_SUB_ELEM	Varchar2(50)			x
MATERIAL_OVERHEAD_SUB_ELEM_ID	Number			x
MATERIAL_SUB_ELEM	Varchar2(50)			x
MATERIAL_SUB_ELEM_ID	Number			x
ORGANIZATION_CODE	Varchar2(3)	*conditionally*		
PROCESS_FLAG	Number	x		
REVISION	Varchar2(3)			x
TRANSACTION_ID	Number		x	
TRANSACTION_TYPE	Varchar2(5)	x		

(Partial List of Columns) Column Name	Type	Required	Derived	Optional
SET_PROCESS_ID	Number	x		

Note: For information about columns that are not discussed in this manual, see the Electronic Technical Reference Manual (eTRM).

Required Data

Every row in the Item Open Interface table must identify the item and organization. To identify the item when importing it, you can specify either the ITEM_NUMBER or SEGMENT*n* columns—the Item Open Interface generates the INVENTORY_ITEM_ID for you. Specifying either the ORGANIZATION_ID or ORGANIZATION_CODE adequately identifies the organization. When more than one of these columns have been entered and they conflict, ITEM_NUMBER overrides SEGMENT*n* and ORGANIZATION_ID overrides ORGANIZATION_CODE. Oracle strongly recommends that you use SEGMENT column instead of ITEM_NUMBER. See: Key Flexfield Segments, *Oracle Flexfields User's Guide*.

Note: If you enter a value for the ITEM_NUMBER column and you are using a multi-segment item, then you must insert the system item flexfield separator between each segment of your item number. For example, if you are using a two segment item and have defined a hyphen (-) as your separator, then enter an item as 1234-5678. When the Item Open Interface derives the segment values of the item, it searches for this separator to indicate the end of one segment value and the start of the next segment value. In our example, 1234 would be put in SEGMENT1, and 5678 in SEGMENT2.

Note: If you enter values for SEGMENT*n* columns, then ensure that the segments you use correspond to the key flexfield segments you defined for your items. No validation for the correct segments occurs when you run the Item Open Interface. Also, the Item Open Interface expects that all segments that you use for the system item flexfield be required segments. Your system items flexfield should not be defined with any optional segments.

Note: No segment validation is done against value sets.

When you import a new item, you must also specify the DESCRIPTION. This must be the same as the master record when you import rows from the child organizations if the description attribute is maintained at the item master level. If the description is at the item-organization level, then you can override the master organization description by giving this column a value.

To manage processing, the Item Open Interface also uses the TRANSACTION_TYPE and PROCESS_FLAG columns. The TRANSACTION_TYPE column tells the Item Open Interface how to handle the row, and the PROCESS_FLAG column indicates the current status of the row.

Set the TRANSACTION_TYPE column to one of the following values:

- **CREATE:** Set this column to CREATE to create an item record (true when both importing a new item and assigning an already existing item to another organization). This is the only value currently supported by the Item Open Interface.

- **UPDATE:** Set this column to UPDATE to update an item category assignment. Enter the category that you want to update in either the Category_Id or Category_Name column in the Item Category Interface table. Enter the existing category that you want to update in the Old_Category_Id or Old_Category_Name column.

- **SYNC:** Transforms into Create if the item otherwise does not exist to update (Item exists).

The Item Open Interface uses the PROCESS_FLAG to indicate whether processing of the row succeeded or failed. When a row is ready to be processed, give the PROCESS_FLAG a value of *1* (Pending) so that the Item Open Interface can pick up the row and process it into the production tables.

Code	Meaning
1	Pending
2	Assign complete
3	Assign/validation failed
4	Validation succeeded; import failed
7	Import succeeded

The preceding table shows a full list of values for the PROCESS_FLAG, but you are unlikely to see all of these.

Other columns, although required in the production tables, are not required in the Item Open Interface table because they have default values or their values can be derived from other sources. Check the defaults and derived values carefully because they might not be the values you want.

If the Item Open Interface successfully processes a row in the Item Open Interface table or the revision interface table, then the program sets the PROCESS_FLAG to **7** (Import succeeded) for the row. If the Item Open Interface cannot insert a row into the production table, then the PROCESS_FLAG column for the failed row is set to **4** (Import failed). If a row in the interface table fails validation, then the PROCESS_FLAG column is set to **3** (validation failed). A row is inserted into the MTL_INTERFACE_ERRORS table for all failed rows. You can review and update any failed rows in each interface table using custom reports and programs.

Derived Data

Many columns have defaults that the Item Open Interface use when you leave that column null in the Item Open Interface table.

(Partial List of Columns) Column Name	Default Value	Value Displayed in Window
SUMMARY_FLAG 1	Y	
ENABLED_FLAG	Y	
PURCHASING_ITEM_FLAG	N	No
SHIPPABLE_ITEM_FLAG	N	No
CUSTOMER_ORDER_FLAG	N	No
INTERNAL_ORDER_FLAG	N	No
SERVICE_ITEM_FLAG	N	No
SERVICE_STARTING_DELAY_DAYS	0	0
INVENTORY_ITEM_FLAG	N	No
ENG_ITEM_FLAG 2	N	No
INVENTORY_ASSET_FLAG	N	No

PURCHASING_ENABLED_FLAG	N	No
CUSTOMER_ORDER_ENABLED_FLAG	N	No
INTERNAL_ORDER_ENABLED_FLAG	N	No
SO_TRANSACTIONS_FLAG	N	No
MTL_TRANSACTIONS_ENABLED_FLAG	N	No
STOCK_ENABLED_FLAG	N	No
BOM_ENABLED_FLAG	N	No
BUILD_IN_WIP_FLAG	N	No
WIP_SUPPLY_TYPE	1	Push
REVISION_QTY_CONTROL_CODE	1	Not under revision quantity control
ALLOW_ITEM_DESC_UPDATE_FLAG	from PO_SYSTEM_PARAMETERS_ALL.ALLOW_ITEM_DESC_UPDATE_FLAG	from Purchasing Options, otherwise Yes
RECEIPT_REQUIRED_FLAG	from PO_SYSTEM_PARAMETERS_ALL.RECEIVING_FLAG	from Purchasing Options, otherwise No

RFQ_REQUIRED_FLAG	from PO_SYSTEM_PARAMETERS_ALL.RFQ_REQUIRED_FLAG	from Purchasing Options, otherwise No
LOT_CONTROL_CODE	1	No lot control
SHELF_LIFE_CODE	1	No shelf life control
SERIAL_NUMBER_CONTROL_CODE	1	No serial number control
RESTRICT_SUBINVENTORIES_CODE	2	Subinventories not restricted to predefined list
RESTRICT_LOCATORS_CODE	2	Locators not restricted to predefined list
LOCATION_CONTROL_CODE	1	No locator control
PLANNING_TIME__FENCE_CODE	4	User-defined time fence
PLANNING_TIME__FENCE_DAYS	1	1
BOM_ITEM_TYPE	4	Standard
PICK_COMPONENTS_FLAG	N	No
REPLENISH_TO_ORDER_FLAG	N	No
ATP_COMPONENTS_FLAG	N	No
ATP_FLAG	N	No
PRIMARY_UNIT_OF_MEASURE	from profile INV: Default Primary Unit of Measure	from Personal Profile Values

ALLOWED_UNITS_LOOKUP_CODE	3	Both standard and item specific
COST_OF_SALES_ACCOUNT	from MTL_PARAMETERS.COST_OF_SALES_ACCOUNT	from Organization Parameters
SALES_ACCOUNT_DSP	from MTL_PARAMETERS.SALES_ACCOUNT	from Organization Parameters
ENCUMBRANCE_ACCOUNT	from MTL_PARAMETERS.ENCUMBRANCE_ACCOUNT	from Organization Parameters
EXPENSE_ACCOUNT	from MTL_PARAMETERS.EXPENSE_ACCOUNT	from Organization Parameters
LIST_PRICE_PER_UNIT	0	0
INVENTORY_ITEM_STATUS_CODE	fromprofile INV: Default Item Status	from Personal Profile Values
INVENTORY_PLANNING_CODE	6	Not planned
PLANNING_MAKE_BUY_CODE	2	Buy
MRP__SAFETY_STOCK_CODE	1	Non-MRP planned
TAXABLE_FLAG	Y	from Purchasing Options, otherwise No
MATERIAL_BILLABLE FLAG	M	Material

EXPENSE_BILLABLE_FLAG	N	No
TIME_BILLABLE_FLAG	N	No
SERVICE_DURATION	0	0
MARKET_PRICE	0	0
PRICE_TOLERANCE_PERCENT	0	0
SHELF_LIFE_DAYS	0	0
RESERVABLE_TYPE	1	Reservable
REPETITIVE_PLANNING_FLAG	N	No
ACCEPTABLE_RATE_DECREASE	0	0
ACCEPTABLE_RATE_INCREASE	0	0
END_ASSEMBLY_PEGGING_FLAG	N	None
POSTPROCESSING_LEAD_TIME	0	0
VENDOR_WARRANTY_FLAG	N	No
SERVICEABLE_COMPONENT_FLAG	N	No
SERVICEABLE_PRODUCT_FLAG	Y	Yes
PREVENTIVE_MAINTENANCE_FLAG	N	No
SHIP_MODEL_COMPLETE	N	No
RETURN_INSPECTION_REQUIREMENT	2	Inspection not required
PRORATE_SERVICE_FLAG	N	No
INVOICEABLE_ITEM_FLAG	N	No

INVOICE_ENABLED_FLAG	N	No
MUST_USE_APPROVED_VENDOR_FLAG	N	No
OUTSIDE_OPERATION_FLAG	N	No
COSTING_ENABLED_FLAG	N	No
CYCLE_COUNT_ENABLED_FLAG	N	No
AUTO_CREATED_CONFIG_FLAG	N	No
MRP_PLANNING_CODE	6	Not planned
CONTAINER_ITEM_FLAG	N	No
VEHICLE_ITEM_FLAG	N	No
END_ASSEMBLY_PEGGING_FLAG	N	None
SERVICE_DURATION	0	0
SET_PROCESS_ID	0	
TRACKING_QUANTITY_IND	P	Primary
ONT_PRICING_QTY_SOURCE	P	Primary
DUAL_UOM_DEVIATION_HIGH	0	0
DUAL_UOM_DEVIATION_LOW	0	0
CONSIGNED_FLAG	2	No
ASN_AUTOEXPIRE_FLAG	2	No
VMI_FORECAST_TYPE	1	Order Forecast
EXCLUDE_FROM_BUDGET_FLAG	2	No
DRP_PLANNED_FLAG	2	No

CRITICAL_COMPONENT_FLAG	2	No
CONTINOUS_TRANSFER	3	Use Global Value
CONVERGENCE	3	Use Global Value
DIVERGENCE	3	Use Global Value
LOT_DIVISIBLE_FLAG	N	No
GRADE_CONTROL_FLAG	N	No
CHILD_LOT_FLAG	N	No
CHILD_LOT_VALIDATION_FLAG	N	No
COPY_LOT_ATTRIBUTE_FLAG	N	No
PROCESS_QUALITY_ENABLED_FLAG	N	No
PROCESS_COSTING_ENABLED_FLAG	N	No
HAZARDOUS_MATERIAL_FLAG	N	No
PREPOSITION_POINT	N	No
REPAIR_PROGRAM	3	Repair Return
OUTSOURCED_ASSEMBLY	2	No
COLLATERAL_FLAG	N	No
EVENT_FLAG	N	No
ELECTRONIC_FLAG	N	No
DOWNLOADABLE_FLAG	N	No
INDIVISIBLE_FLAG	N	No

Notes:

1 Defaulted to Y by the Item Open Interface, but the Master Items window defaults N for this column.

2 Defaulted to N by the Item Open Interface, but in the Master Items window the default value depends on whether the window is accessed from Oracle Engineering.

You can import item descriptive flexfield values if you implemented a descriptive flexfield for items. To do this, include values for the descriptive flexfield columns (ATTRIBUTE_CATEGORY and ATTRIBUTEn columns) in the Item Open Interface table. No validation is performed on descriptive flexfield values.

In addition, the Item Open Interface uses the status of the item (INVENTORY_ITEM_STATUS_CODE) to determine the value of attributes under status control. If an attribute is under status control, then the attribute value always derives from the status of the item, and any value in the attribute column of the Item Open Interface table is ignored. If an attribute is under default status control, then the attribute value derives from the status of the item only if no value is in the attribute column of the Item Open Interface table. If an attribute is not under any status control, then the item status has no effect on the value of the attribute for the imported item.

> **Note:** If an attribute is under status control, then it still must follow the attribute dependency rules. For example, if the BOM_ENABLED_FLAG is under status control, and a status is used setting BOM_ENABLED_FLAG to Yes, then the INVENTORY_ITEM_FLAG must be set to Yes for the imported item. If the item has INVENTORY_ITEM_FLAG set to No (or it is left null and therefore defaults to No), then the Item Open Interface processes the item with the BOM_ENABLED_FLAG set to No. This is because the attribute dependency rules stipulate that BOM_ENABLED_FLAG can be only Yes for an Inventory Item.

> **Note:** When you assign an item to a child organization, all item-level attributes default down from the master organization, but only when the attribute column is null in the Item Open Interface table. If you supply a value for an item-level attribute in a child organization record, then the Item Open Interface rejects the record as an error. The exception is status attributes under status control. These attributes *always* derive from the status of the item, never from the master record.

Whether you import a new item to a master organization or assign an existing item to a nonmaster organization, the Item Open Interface always enters a unique numeric identifier in the TRANSACTION_ID column of the Item Open Interface table, and the concurrent request number in the REQUEST_ID column of the item master table.

Item Categories

When the Item Open Interface imports an item, it also assigns the item to the

mandatory category sets based on the item defining attributes. The default category for each category set is used. The Item Open Interface also lets you assign items to other category sets and categories when data exists for item category assignments in the MTL_ITEM_CATEGORIES_INTERFACE table. See: Defining Category Sets, *Oracle Inventory User's Guide* and Defining Default Category Sets, *Oracle Inventory User's Guide*.

For example, suppose that you define a category set *Inventory* with a default category of *glass,* and you designate *Inventory* as the mandatory category set for inventory items. When the interface imports an inventory item (INVENTORY_ITEM_FLAG = *Y*), the item is assigned to the *glass* category in the *Inventory* category set, and a corresponding row is inserted into MTL_ITEM_CATEGORIES.

When you are using the Item Open Interface to assign an existing item to another organization, the item is also assigned to mandatory category sets with the default category. As described previously, the item-defining attributes determine to which mandatory category sets the item is assigned. Even if the item is assigned to an item-level category set (nonmandatory) in the master organization, it is not assigned to that category set in the new organization of the item.

> **Note:** When you are running Import Items in Update mode, if the defining attribute for a functional area is enabled, then the proper default category set and category are assigned to the item.

Validation

When you import or update an item, the Item Open Interface validates the data and any derived values the same way that manually entered items are validated. This validation ensures that:

- Required columns have an included or defaulted value.
- Control levels are reflected in item attribute values.
- Status control settings for status attributes are maintained.
- Interdependences between item attribute values are consistent.

> **Note:** Before you can import an item into a child organization, it must exist in the master organization. The master records are automatically separated from child records, and processed first. However, because one Item Open Interface process is not aware of other Item Open Interface processes running in parallel, do not split separate organization records of a given item into two different set_process_id that are running in parallel. This could cause the child record to be processed before the master record. See: Defining Items, *Oracle Inventory User's Guide*

When you import items, the Item Open Interface program validates all rows in the table that have a PROCESS_FLAG set to 1 (Pending). The interface first assigns the default values to the derived columns of the row, then updates the value of the PROCESS_FLAG column to 2 (Assign Succeeded).

The Item Open Interface then validates each row. If a row in the interface table fails validation, then the program sets the PROCESS_FLAG to 3 (Assign/Validation Failed) and inserts a row into the error table.

For all successfully validated rows, the interface inserts the rows into the Oracle Inventory item master table, MTL_SYSTEM_ITEMS. If a row cannot be inserted into the item master table, then the program sets the PROCESS_FLAG to 4(Import Failed) and inserts a row into the error table.

After this program inserts the imported item into the item master table, the row is deleted or, depending on the run-time option, the PROCESS_FLAG is set to 7 (Import Succeeded).

To minimize the number of rows that are stored in the interface table, you can specify at run time that the program delete successfully processed records after insertion. If you do not delete successfully processed records automatically, then you can write custom programs that report and delete any successfully imported rows. The program can search for rows with a PROCESS_FLAG value of 7 (Import Succeeded), list the rows in a report, and then delete them from the table. By defining a report set in Oracle Application Object Library, you can automatically run the custom program after each submission of the Item Open Interface. You can also run multiple Item Open Interface processes. See: Multithread Capability subsequently.

Importing Additional Item Details

You can import additional cost and revision details for an imported item using interface tables that are listed in the following table. The Item Open Interface imports the details that are specified in these tables at the same time that it imports the items themselves. The program validates all rows that you insert into the interface tables, derives additional columns, and creates the item and item details in Oracle Inventory.

Item Detail	Interface Table	Number of Rows per Item
Costs	MTL_SYSTEM_ITEMS_INTERFACE	1
Revisions	MTL_SYSTEM_ITEMS_INTERFACE *(for imported items only)*	1
	MTL_ITEM_REVISIONS_INTERFACE	1 or more

Note: Although many other tables are in Oracle Inventory with names that imply use, the tables listed in the table are the *only* interface tables that are used by the Item Open Interface to import item details.

Before importing additional item details, you must complete the same setup steps that are required for manually defining these item details. For example, you must define your cost types and activities before you can assign item costs. You must specify the default cost category set using the Default Category Sets window, and set the starting revision for all organizations. See: Overview of Setting Up, *Oracle Inventory User's Guide*, Defining Default Category Sets, *Oracle Inventory User's Guide*, and Defining Item Revisions, *Oracle Inventory User's Guide*.

The Item Open Interface validates all required data and some optional data that is included in the item detail interface tables. When you import your cost or revision data, this program validates the included or derived values the same way that Oracle Inventory validates manually entered details.

Importing Cost Details

When the Item Open Interface imports an item, it can also import costing information into Oracle Cost Management tables. The interface can import this costing information automatically using organization and category defaults, or you can specify the information for the item itself in the Item Open Interface table.

If you set up a default material overhead rate for the organization of an item or for the default cost category, then this material overhead rate is inserted into the cost details table, CST_ITEM_COST_DETAILS, and summarized in the item costs table**,** CST_ITEM_COSTS. See: Defining Material Subelements, *Oracle Cost Management User's Guide* and Defining Overhead, *Oracle Cost Management User's Guide*.

You can specify one material cost and one material overhead rate for the item directly in the Item Open Interface table itself. Remember to include the material subelement for the material cost and overhead rate by specifying the subelement.

The interface imports the basis type for the material subelements and material overhead subelements from the BOM_RESOURCES table. If the default basis type is not defined, then the basis type of these subelements is set to *Item* and *Total Value,* respectively.

Note: When you run the Item Open Interface in Update mode, you cannot update item costs.

Importing Revision Details

You can import detailed revision history with your new items in any one of the following ways:

- Specify revisions and effectivity dates in the revision interface table.

- Specify the current revision for each item in the Item Open Interface table.
- Do not specify any revisions and let the Item Open Interface default the revision based on the starting revision that is defined in the Organization Parameters window. See: Organization Parameters Window, *Oracle Inventory User's Guide*.

To import multiple item revisions and effectivity dates, use the revision interface table, MTL_ITEM_REVISIONS_INTERFACE. You can also include engineering change number (ECN) information. You need to create your own program for populating this table.

Because revisions exist at the item-organization level, you need revision data for each item-organization you are updating. Include a row for each revision (with an effectivity date) to import, in ascending order. In other words, for each item-organization combination, revision A must have an effectivity date that is less than or equal to revision B, and so on. Each row in this table must correspond to a row in the Item Open Interface table. Each row must reference the ITEM_NUMBER and ORGANIZATION_ID or ORGANIZATION_CODE for the item.

> **Note:** When you import multiple revisions for the same item, if one of the revisions fails validation, then all revisions for that item fail.

To import an item and its current revision only, include a value for the REVISION column in the Item Open Interface table. The Item Open Interface automatically creates this revision with an effective date that is equal to the system date when it imports the item. (Use the revision interface table described previously if you want to specify a revision effectivity date.)

If you choose not to use the revision interface table and do not include a revision in the Item Open Interface table, then the Item Open Interface assigns each item a beginning revision, using the default that is specified in the Organization parameters. The system date is the effectivity date. After the beginning revision effectivity date is established, you cannot add revisions with effectivity dates earlier than the date assigned by the Item Open Interface.

> **Note:** Although most item information defaults from the master organization when you assign an existing item to a child organization, the Item Open Interface does *not* default the revision detail of an item from the master organization. See: Defining Item Revisions, *Oracle Inventory User's Guide*.

As with the Item Open Interface table, the column PROCESS_FLAG indicates the current state of processing for a row in the revision interface table. Possible values for the column are listed in the following table.

When you insert rows into the revision interface table, set the PROCESS_FLAG to 1 (Pending) and TRANSACTION_TYPE to CREATE.

MTL_ITEM_REVISIONS Column Name	MTL_ITEM_REVISIONS_INTERFACE Column Source
INVENTORY_ITEM_ID	ITEM_NUMBER
ORGANIZATION_ID	ORGANIZATION_ID or ORGANIZATION_CODE
REVISION	REVISION
CHANGE_NOTICE	CHANGE_NOTICE
ECN_INITIATION_DATE	ECN_INITIATION_DATE
IMPLEMENTATION_DATE	IMPLEMENTATION_DATE
IMPLEMENTED_SERIAL_NUMBER	IMPLEMENTED_SERIAL_NUMBER
EFFECTIVITY_DATE	EFFECTIVITY_DATE
ATTRIBUTE_CATEGORY	ATTRIBUTE_CATEGORY
ATTRIBUTE*n*	ATTRIBUTE*n*
REVISED_ITEM_SEQUENCE_ID	REVISED_ITEM_SEQUENCE_ID
DESCRIPTION	DESCRIPTION

You can import revision descriptive flexfield values when you have implemented a descriptive flexfield for revisions. To do this, include values for the descriptive flexfield columns (ATTRIBUTE_CATEGORY and ATTRIBUTE*n* columns) in the revision interface table when you import revisions.

The Item Open Interface can also be used to create revisions for existing items. Only revision labels and effectivity dates that are higher than existing revisions can be imported. To do this, load revision detail for the existing items into MTL_ITEM_REVISIONS_INTERFACE and run the Item Open Interface.

> **Note:** You cannot add new item revisions to existing items while running Import Items in Update mode.

Importing Item Category Assignments

When the Item Open Interface imports items, it can also import item category assignments into Oracle Inventory. For this to happen, you must insert item category assignments data into the MTL_ITEM_CATEGORIES_INTERFACE table

> **Note:** Item category assignments are imported when you run the Item Open Interface in either the Create mode or Update mode.

You can also import item category assignments as a separate task by selecting Import Item Category Assignments from the Import submenu of the Inventory menu. In this case, items are not imported.

Item Category Assignment Interface Runtime Options

To run the Item Open Interface, select Import Item Category Assignments from the Import submenu of the Inventory menu.

When you run the Item Category interface, you are prompted for report parameters. These are run-time options for the Item Category interface:

Record Set ID

Enter a set ID number for the set of records in the interface table that you want to process. The program picks up the records that have that ID in the SET_PROCESS_ID column. This column value cannot be NULL. To avoid set ID, value conflicts between record sets that are imported by different users, should use the sequence MTL_SYSTEM_ITEMS_INTF_SETS_S when populating the SET_PROCESS_ID column.

Upload Processed Records

Yes: All qualifying item category assignment records in the interface table are inserted into Oracle Inventory.

No: Do not insert item category assignments into Oracle Inventory. Use this option if you want to validate items without any processing.

Delete Processed Records

Yes: Delete successfully processed item category assignment records from the Item Categories interface table.

No: Leave all records in the Item Categories interface table.

> **Note:** In contrast to the transaction type control in the Item Open Interface, the Item Categories Open Interface processes all transaction types (CREATE, DELETE) in the current record set, as specified in the

TRANSACTION_TYPE column, and it is not controlled by run-time option (parameter).

Inserting Item Category Information into the Item Categories Interface Table

Item Categories Interface Description

The Item Category interface table, MTL_ITEM_CATEGORIES_INTERFACE, contains every column in the Oracle Inventory Item Category Assignments table, MTL_ITEM_CATEGORIES. The columns in the Item Category interface table correspond directly to those in the Item Category Assignments table. In addition, transaction control columns and attribute value columns exist that correspond to the ID columns. These are described subsequently.

Before you import item category assignments, you must populate the following MTL_ITEM_CATEGORIES_INTERFACE columns:

- Transaction Control Columns
- SET_PROCESS_ID: SET_PROCESS_ID can be envisaged as a record set to be processed by a single import program run. This value ties the categories interface rows back to the MTL_SYSTEM_ITEMS_INTERFACE rows when item category assignments are imported together with items. To prevent interference with other users' interface data, select a SET_PROCESS_ID value from the MTL_SYSTEM_ITEMS_INTF_SETS_S sequence.
- TRANSACTION_TYPE: Valid values are CREATE and DELETE. To update a transaction, you must delete the assignment and then create it again with the change you want.
- PROCESS_FLAG: Indicates current status of a record (initially = 1). Attributes that define a particular assignment to be imported. You can provide them either as values or IDs:
  - ORGANIZATION_ID or ORGANIZATION_CODE
  - INVENTORY_ITEM_ID or ITEM_NUMBER
  - CATEGORY_SET_ID or CATEGORY_SET_NAME
  - CATEGORY_ID or CATEGORY_NAME

Update Existing Items

You can run Import Items in Update mode by setting the Create or Update Items parameter to 2. You can update existing item attributes, and assign templates to existing

items.

Every item attribute can be updated, and all necessary validations are performed to enforce item attribute interdependencies, master/child attribute dependencies, and status controlled dependencies.

Attributes that are controlled at the master level are correctly propagated to the child records when the Master Item record is updated. The child records are copied into MTL_SYSTEM_ITEMS_INTERFACE for validation and are identified with a transaction_type, AUTO_CHILD. These records are deleted if the Delete Processed Rows parameter has been passed as Yes and remain for diagnostic purposes if the parameter is passed as No.

The status can be changed for existing items and the proper attributes default accordingly. The status change is then recorded in the MTL_PENDING_ITEM_STATUS table.

When the defining attribute for a functional area is enabled, the proper default category set and category is assigned to the item.

> **Note:** If another user or process locked a given item record, then the update for the item in all organizations produces an error in order to maintain master/child relationship integrity. In this case, the transaction_id column populates with the inventory_item_id of the locked record, within the row written to the MTL_INTERFACE_ERRORS table.

Determining Batch Size

Running the Item Open Interface in Create Mode

The optimum batch size depends upon the amount of data being imported and the power of your system. If you are processing 10,000 items or less, then break the 10,000 items into smaller groups. A batch size of 1,000 to 2,000 should be adequate to process these rows efficiently. Load the records for one group, process them, and then delete the records from the interface table. If you want to maintain an interface row history, then create separate history tables and move processed rows there instead of deleting them. Leaving large numbers of processed rows in the interface table degrades performance of the Item Open Interface.

For a greater number of records, you can also use a batch size of 2,000, but you may want to try larger batch sizes, up to 5,000 items. The best way to determine your batch size is by benchmarking.

First, try loading 2,000 items into the interface tables. Note the time that the Item Open Interface takes to process these rows in one batch. If this time is more than 20 minutes, then use a batch size of 2,000 or less. Otherwise, try larger batch sizes.

If you want to import a large number of records, then break down your records into

batch size, as described previously, and process them in parallel for optimum performance. See: Multithread Capability (Parallel Runs of the Item Open Interface), for more information.

> **Note:** If you are using batch management, then do not delete rows from the interface tables.

Running the Item Open Interface in Update Mode

When you run the Item Open Interface in Update mode, the proper batch size is approximately five times smaller than the batch size from running the Item Open Interface in Create mode. This is due to the extra processing that is required for child records. For example, if you are updating 100 master records, and each item is assigned to four organizations, then the Item Open Interface processes the 100 populated records, plus the 400 additional, automatically created, AUTO_CHILD (transaction_type within the MTL_SYSTEM_ITEMS_INTERFACE table) records in the interface table.

Resolving Failed Interface Rows

If a row fails validation, then the Item Open Interface sets the PROCESS_FLAG to 3 (Assign/validation failed) and inserts a row in the interface errors table, MTL_INTERFACE_ERRORS. To identify the error message for the failed row, the program automatically populates the TRANSACTION_ID column in this table with the TRANSACTION_ID value from the corresponding Item Open Interface table. For example, if a row in the Item Open Interface table fails, then the program inserts a row into the interface errors table with the TRANSACTION_ID of the failed row as the TRANSACTION_ID of the error row. Each error in the interface errors table has a value for the MESSAGE_NAME and REQUEST_ID columns. The Item Open Interface populates these columns for item detail errors the same way it populates the table for item errors.

The UNIQUE_ID column in MTL_INTERFACE_ERRORS is populated from the sequence MTL_SYSTEM_ITEMS_INTERFACE_S. Thus, for a given row, the sequence of errors can be determined by examining UNIQUE_ID for a given TRANSACTION_ID.

For example, if your row with TRANSACTION_ID 2000 failed with two errors in the MTL_INTERFACE_ERRORS table, then you can see which error occurred first by looking at the row with the smallest UNIQUE_ID for TRANSACTION_ID 2000.

Resolve errors in the sequence in which they were found by the interface, that is, in increasing order of UNIQUE_ID for any TRANSACTION_ID.

Quite often, resolving the first few errors and restarting the Item Open Interface causes the other (spurious) errors for that failed row to disappear. However, several conditions that previously caused the interface to error all rows have been modified so that only the offending row is marked in error, while the remaining rows process normally.

The Item Open Interface inserts validated rows into the production tables in Oracle

Inventory and Oracle Cost Management. Depending on the item information that you import, the interface inserts these rows into the item master, item categories, item costs, cost details, item revisions, pending item status, and unit of measure conversion tables. If a row cannot be inserted into one of these tables, then the PROCESS_FLAG column for all remaining rows is set to 4 (Import failed) and the Concurrent Item Open Interface inserts a row in the interface errors table. The program handles these processing errors in the same way that it handles validation errors.

Reviewing Failed Rows

You can review and report rows in any of the interface tables using SQL*Plus or any custom reports that you develop. Because all rows in the interface tables have a value for PROCESS_FLAG, you can identify records that were not successfully imported into Oracle Inventory.

Resubmitting an Errored Row

During Item Open Interface processing, rows can error out either due to validation (indicated by PROCESS_FLAG = 3 in MTL_SYSTEM_ITEMS_INTERFACE and the corresponding error in MTL_INTERFACE_ERRORS) or due to an Oracle error.

When an Oracle error is encountered, the processing is stopped and everything is rolled back to the previous save point. This could be at PROCESS_FLAG = 1, 2, 3, or 4.

PROCESS_FLAG	Error After and Before PROCESS_FLAG	Relaunch the Item Open Interface after
1	1 and 2	Fixing the Oracle error
2	2 and 3	Fixing the Oracle error
3	2 and 3	Updating MSII[1] and fixing the corresponding error in MIE[2]. Then setting PROCESS_FLAG = 1 and INVENTORY_ITEM_ID = null in MSII[1], MICI[3], and MIRI[4].
4	4 and 7	Fixing the Oracle error

1. MTL_SYSTEM_ITEMS_INTERFACE
2. MTL_INTERFACE_ERRORS
3. MTL_ITEM_CATEGORIES_INTERFACE

4. MTL_ITEM_REVISIONS_INTERFACE

When you encounter rows that errored out due to validations, you must first fix the row that corresponds to the error with the appropriate value. Reset PROCESS_FLAG = 1, INVENTORY_ITEM_ID = null, and TRANSACTION_ID = null. Resubmit the row for reprocessing.

Multithread Capability (Parallel Runs of the Item Open Interface)

The following tables have a NOT NULL NUMBER column called SET_PROCESS_ID:

- MTL_SYSTEM_ITEMS_INTERFACE
- MTL_ITEM_REVISIONS_INTERFACE
- MTL_ITEM_CATEGORIES_INTERFACE

The SET_PROCESS_ID column has a database default value of zero in the three tables preceding.

To have parallel runs of the Item Open Interface, populate the SET_PROCESS_ID column for records in the interface tables with a positive, nonzero number; you can enter any SET_PROCESS_ID between 1 and 2,147,483,647. If you enter a value greater than this, then that interface record is not processed.

Assign all organization records of the same item to the same set_process_id.

Example

You have 1000 records in the MTL_SYSTEM_ITEMS_INTERFACE table that you want to insert into MTL_SYSTEM_ITEMS, and you decide to have four parallel Item Open Interface processes to accomplish this task.

In the scripts that you use to insert data into the MTL_SYSTEM_ITEMS_INTERFACE table, populate the first 250 records with SET_PROCESS_ID = 1, the next 250 records with SET_PROCESS_ID = 2, and so on. Oracle recommends dividing your records into roughly equal batch sizes.

> **Note:** If you have custom scripts to insert data into the MTL_SYSTEM_ITEMS_INTERFACE table, then you must modify them to include the SET_PROCESS_ID column. Also remember that any corresponding records that you enter in the MTL_ITEM_REVISIONS_INTERFACE table must have matching SET_PROCESS_ID values.

From the Item Open Interface SRS launch form, specify the Process Set for a given run in the Process Set parameter. This initiates four Item Open Interface concurrent programs with the Process Set parameter set to 1, 2, 3, and 4, respectively. The four Item Open Interface processes run in parallel, each working on the set that you specified.

Note: Leaving a null in the Process Set parameter will process all rows, regardless of the process set value. If you do not want the new multithread capability, then you can populate the interface tables as you always have. The SET_PROCESS_ID column will get the default value of zero. When you run the Item Open Interface with the Process Set parameter blank, the interface processes all rows (regardless of the SET_PROCESS_ID value) as it did in earlier releases.

Multithreading Rules

Oracle recommends that the applications database administrator for the site enforce these rules:

- If you run an Item Open Interface process with the Process Set value null, then you should not run any other Item Open Interface processes.
- Do not have parallel runs of the Item Open Interface on the same process set.
- Do not create an enormous number of parallel processes. All processes must access the same tables; increasing the number of processes does not always increase throughput.
- Do not use the same set_process_id for the records that you ran in Create mode as the records you ran in Update mode.

If you do not follow these rules, then multiple Item Open Interface processes will attempt to work on the same set of rows, which could lead to unpredictable errors.

Concurrent Requests Used To Facilitate Parallel Processing

Use the Import Items request to facilitate parallel processing. Access this request using the Item Reports and All Reports menu options, within the Inventory Responsibility. The request launches five INCOIN processes in parallel that either create or update items, depending on the parameter settings.

Note: Using the System Administrator Responsibility, you can modify the request to launch a greater or lesser number of parallel processes. (Navigator > Request > Set)

Parameters

When you run the request in Update mode, the request calls INCOIN with five parallel sessions. The parameters default in as follows:

- All Organizations
- Yes

- Validate Items
- Yes
- Process Items
- Yes
- Delete Processed Rows
- Yes
- Process Set
- 101, 102, 103, 104, 105
- Create or Update Items
- 1

Default Parameters

When you run the request in Create mode, the request calls INCOIN with five parallel sessions. The parameters default in as follows:

- All Organizations
- Yes
- Validate Items
- Yes
- Process Items
- Yes
- Delete Processed Rows
- Yes
- Process Set
- 201, 202, 203, 204, 205
- Create or Update Items
- 2

Category Application Program Interface

This section explains how to use the Category API (INV_ITEM_CATEGORY_PUB) and how the API functions in Oracle Inventory and Oracle Product Lifecycle Management.

The PL/SQL category API helps you maintain categories, valid categories, and category assignments to an item.

Category API Features

The Category API has different procedures to handle the CREATE, UPDATE, or DELETE transaction types.

Tables involved:

- mtl_category_sets_b
- mtl_categories_b
- mtl_categories_tl
- mtl_category_set_valid_cats
- mtl_item_categories

The Category API also defines record CATEGORY_REC_TYPE, which is part of the argument list of most of the calls to this API.

- mtl_categories_b/tl: Stores the list of categories.
- mtl_category_sets_b: Stores the list of category sets.
- mtl_categori_set_valid_cats: Stores the list of valid categories for a category set which has enable list of valid categories property value as true.
- mtl_item_categories: Stores the list of associations of item and categories.

Functional Overview

The Category API supports the CREATE, UPDATE, AND DELETE transaction types on the following entities:

- Categories
- Item categories
- Valid categories

Setting Up the Category API

You must initialize the following parameters before calling this PL/SQL API:

- Version
- Category_rec_type record

Parameter Descriptions

The following table lists all parameters used by the public Category API. All of the inbound and outbound parameters are listed. The following table contains additional information about these parameters.

The following table contains a list of calls available in this API. The list of parameters is discussed in greater detail at the end of group of similar calls. You can also find details about the Category API on Oracle Integration Repository.

Create_category - Use this API to create a category.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	YES	
X_MSG_DATA	OUT	VARCHAR2	YES	
P_CATEGORY_REC	IN	INV_ITEM_CATEGORY_PUB.CATEGORY_REC_TYPE	Yes	
X_CATEGORY_ID	OUT	NUMBER	Yes	

Update_category - Use this API to update a category.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	YES	
X_MSG_DATA	OUT	VARCHAR2	YES	
P_CATEGORY_REC	IN	INV_ITEM_CATEGORY_PUB.CATEGORY_REC_TYPE	Yes	

Delete_category - Use this API to delete a category.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	Yes	
X_MSG_DATA	OUT	VARCHAR2	Yes	

P_CATEGORY_REC	IN	INV_ITEM_CATEGORY_PUB.CATEGORY_REC_TYPE	Yes
P_CATEGORY_ID	OUT	NUMBER	Yes

Refer to the INV_ITEM_CATEGORY_PUB specification for record structure details of P_CATEGORY_REC.

The parameter description is as follows:

- P_API_VERSION: A decimal number that indicates major and minor revisions to the API. Pass 1.0 unless otherwise indicated in the API parameter list.
- P_INT_MSG_LIST: A one-character flag that indicates whether to initialize the message stack of the FND_MSG_PUB package at the beginning of API processing. This removes any messages that exist on the stack from prior processing in the same session. Valid values are FND_API.G_TRUE and FND_API.G_FALSE. The default value is FND_API.G_FALSE.
- P_COMMIT: A one-character flag that indicates whether to commit work at the end of API processing. Valid values are FND_API.G_TRUE and FND_API.G_FALSE. The default value is FND_API.G_FALSE.
- X_RETURN_STATUS: A one-character code that indicates whether any errors occurred during processing. If an error occurs, then case error messages are present on the message stack of the FND_MSG_PUB package. Valid values are FND_API.G_RET_STS_SUCCESS, FND_API.G_RET_STS_ERROR, and FND_API.G_RET_STS_UNEXP_ERROR.
- X_MSG_COUNT: An integer that indicates the number of messages on the message stack of the FND_MSG_PUB package at the end of API processing.
- X_MSG_DATA: A character string that contains message text. This character string is not considered empty only when x_msg_count is greater than 1. This feature makes it possible for callers to avoid interacting with the message stack when there is only one error message, as is commonly the case.
- P_category_id: Holds the category_id for the entity row that was modified or created.

Create_valid_category - Use this API to create valid categories of a category set.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
P_CATEGORY_SET_ID	IN	NUMBER	Yes	
P_CATEGORY_ID	IN	NUMBER	Yes	
P_PARENT_CATEGORY_ID	IN	NUMBER	Yes	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	Yes	
X_MSG_DATA	OUT	VARCHAR2	Yes	

Update_valid_category - Use this API to update valid categories of a category set.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
P_CATEGORY_SET_ID	IN	NUMBER	Yes	
P_CATEGORY_ID	IN	NUMBER	Yes	
P_PARENT_CATEGORY_ID	IN	NUMBER	Yes	

X_RETURN_STATUS	OUT	VARCHAR2	Yes
X_ERROR_CODE	OUT	NUMBER	Yes
X_MSG_COUNT	OUT	NUMBER	Yes
X_MSG_DATA	OUT	VARCHAR2	Yes

Delete_valid_category - Use this API to delete valid categories from a category set.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
P_CATEGORY_SET_ID	IN	NUMBER	Yes	
P_CATEGORY_ID	IN	NUMBER	Yes	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	Yes	
X_MSG_DATA	OUT	VARCHAR2	Yes	

The parameters description is as follows:

- P_category_id: Contains the category ID.
- P_category_set_id: Identifies the category set.
- P_parent_category_id: Identifies the parent of the current category for the Update call. For create, if the value is NULL, then this category becomes a first-level child in the category set.

Validation of Category ID

Standard Validation

Oracle Inventory validates all required columns in the Category API. For specific information on the data referenced by these columns, see the Electronic Technical Reference Manual (eTRM) for details.

The API validates the category ID that is passed to it. category_set_id is also checked to see if it allows valid categories. For update, a category is valid if the parent and child category IDs are not NULL and are different from each other.

Other Validation

The Category API verifies that the required parameters are not passed as NULL. For hierarchical category sets, the Category API checks that a loop is not introduced due to a CREATE or UPDATE operation. You cannot delete categories that are default categories or that have items assigned to them.

Error Handling

If any validation fails, the Category API returns the error status to the calling module. The API processes the rows and reports the following values for every record:

Condition	PROCESS_STATUS	ERROR_MESSAGE
Success / Failure	4 / 5	null / actual error message
Failure		INV_MISSING_REQUIRED_PARAMETER
Failure	N/A	INV_SAME_CATEGORY_SETS
Failure	N/A	INV_CATEGORY_UNAVAIL_UPDATE
Failure	N/A	INV_CATEGORY_LOOPS_ERR
Failure	N/A	INV_MISSING_PARENT_CAT
Failure	N/A	INV_DEFAULT_CATEGORY_ADD_ERR
Failure	N/A	INV_UNWANTED_PARENT_CAT

Item Category Assignment API

Item Category Assignment API Features

Use the Item Category Assignment API to assign items [mtl_system_items_b] to categories [mtl_categories_b]. The assignments are stored in mtl_item_categories table.

Functional Overview

The Item Category Assignment API supports CREATE, DELETE, and UPDATE of item category assignments. Before you create an assignment, you must first define the item and category (category_set).

Setting Up the Item Category Assignment API

The setup is identical to the setup for the Category API.

Parameter Descriptions

The following table describes all parameters that are used by the public Item Category Assignment API. The table lists all of the inbound and outbound parameters. After the table, additional information is provided.

The various calls are detailed below.

Create_category_assignment - Use this API to create an item category assignment.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	YES	
X_MSG_DATA	OUT	VARCHAR2	YES	

P_CATEGORY_SET_ID	IN	NUMBER	Yes	
P_CATEGORY_ID	IN	NUMBER	Yes	
P_INVENTORY_ITEM_ID	IN	NUMBER	Yes	
P_ORGANIZATION_ID	IN	NUMBER	Yes	

Update_category_assignment- Use this API to reassign an item to a new category in the given category set.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	YES	
X_MSG_DATA	OUT	VARCHAR2	YES	
P_CATEGORY_SET_ID	IN	NUMBER	Yes	
P_CATEGORY_ID	IN	NUMBER	Yes	
P_OLD_CATEGORY_ID	IN	NUMBER	Yes	
P_INVENTORY_ITEM_ID	IN	NUMBER	Yes	
P_ORGANIZATION_ID	IN	NUMBER	Yes	

Delete_category_assignment - Use this API to delete a item category assignment.

Parameter Name	Usage	Type	Required?	Derived?
P_API_VERSION	IN	NUMBER	Yes	
P_INIT_MSG_LIST	IN	VARCHAR2	No	
P_COMMIT	IN	VARCHAR2	No	
X_RETURN_STATUS	OUT	VARCHAR2	Yes	
X_ERROR_CODE	OUT	NUMBER	Yes	
X_MSG_COUNT	OUT	NUMBER	YES	
X_MSG_DATA	OUT	VARCHAR2	YES	
P_CATEGORY_SET_ID	IN	NUMBER	Yes	
P_CATEGORY_ID	IN	NUMBER	Yes	
P_INVENTORY_ITEM_ID	IN	NUMBER	Yes	
P_ORGANIZATION_ID	IN	NUMBER	Yes	

The parameters description is as follows. Parameters shared with the Category API are described above.

- P_inventory_item_id: Contains the item ID of the item to which the category is being assigned.
- P_organization_id: Identifies the organization to which the item belongs.
- P_old_category_id: Identifies the current category to which the item is assigned. It is only used for updating the category assignment.

Validation of Item Category Assignment API

Standard Validation

Oracle Inventory validates all required columns in the Item Category Assignment API.

For specific information on the data referenced by these columns, see the Electronic Technical Reference Manual (eTRM) for details.

The Item Category Assignment API checks for NULL values in the required parameters in addition to the other validations listed below. It also validates that the category and item primary keys passed are valid and the approval status of the item does not disallow changes.

Creating Assignment Set Entries

When you create new entries, the following validations occur:

- Multiple category assignments are not made in a category set that does not allow it.
- Master-controlled category sets are not created in a child organization context.
- For valid list-enabled category sets, the category to which an item is being assigned exists in the list.
- For hierarchical catalogs, assignments are only made to leaf categories in a hierarchy.

If validation is successful, then a record is inserted into the mtl_item_categories table.

Updating Assignment Set Entries

When you update an existing entry, the following validations occur:

- The assignment exists.
- The new category belongs to a valid set and is a leaf node, if the catalog mandates it.

If validation is successful, then a record in mtl_item_categories table is modified.

Deleting Assignment Set Entries

When you delete an existing assignment set, the following validations occur:

- The default category assignment for a functional area is not deleted.
- The assignment exist.

The assignment set record is deleted from mtl_item_categories table.

Error Handling

If any validation fails, then the Item Category Assignment API returns the error status to the calling module. The API processes the rows and reports the following values for every record:

Condition	PROCESS_STATUS	ERROR_MESSAGE
Success / Failure	4 / 5	null / actual error message
Failure	N/A	INV_INVALID_ARG_NULL_VALUE
Failure	N/A	INV_ORGITEM_ID_NOT_FOUND
Failure	N/A	INV_IOI_NIR_NOT_COMPLETE
Failure	N/A	INV_CATEGORY_SET_ID_NOT_FOUND
Failure	N/A	INV_CATEGORY_ID_NOT_FOUND
Failure	N/A	INV_INVALID_CATEGORY_STRUCTURE
Failure	N/A	INV_CAT_CANNOT_CREATE_DELETE
Failure	N/A	INV_CATEGORY_NOT_IN_VALID_SET
Failure	N/A	INV_ITEM_CAT_ASSIGN_LEAF_ONLY
Failure	N/A	INV_ITEM_CAT_ASSIGN_NO_MULT
Failure	N/A	INV_CAT_ASSGN_NOT_FOUND

Customer Item and Customer Item Cross-Reference Open Interfaces

Some manufacturing industries use a multi-tiered, just-in-time supply chain structure. Today's manufacturing environment requires a close working relationship between customers and suppliers along the entire supply chain. Suppliers must react quickly to their customers' often changing requirements. By cross-referencing customer items with their own inventory items, suppliers can achieve faster order processing and shipments by allowing customers to place orders using customer item numbers.

You can import customer items and customer item cross-references from any legacy system into Oracle Inventory using the Customer Item Open Interface and the Customer Item Cross-Reference Interface. These interfaces validate all data that you import into Oracle Inventory. They also validate foreign keys and check for attribute interdependencies, acceptable values, and value ranges. The interfaces ensure that the imported customer items and cross-references contain the same detail as items that are entered manually using the Customer Items and Customer Item Cross-References

windows. Error codes and corresponding error messages for all errors detected during validation are written to the interface tables.

Functional Overview - Customer Item Open Interface

The Customer Item Open Interface lets you import customer items into Oracle Inventory. For each customer item, you must define related information such as the customer and item definition level. In addition, you can provide master and detail container information, commodity codes, model items, and other attributes such as demand tolerances and Departure Planning flags for each customer item. See: Defining Customer Items, *Oracle Inventory User's Guide*.

After you add new customer items to the MTL_CI_INTERFACE table, run the Customer Item Open Interface. The Customer Item Open Interface reads each record from the interface table and adds items that are successfully validated to the MTL_CUSTOMER_ITEMS table. When customer items are validated, the same rules are used as for the Customer Items window to ensure that only valid items are imported.

Functional Overview Customer Item Cross-Reference Interface

The Customer Item Cross-Reference Interface lets you import cross-references between customer items and existing Oracle Inventory items into your master organization. For each customer item cross-reference, you must define the customer, customer item, customer item definition level, and rank. Create a cross-reference to the associated Oracle Inventory item by specifying the item and its master organization.

You can create multiple cross-references between customer items and one Oracle Inventory item. You can also create multiple cross-references between Oracle Inventory items and one customer item. Cross-references are defined at the master organization level of the cross-referenced inventory item. After you define a customer item cross-reference to an Inventory item, the cross-reference is applicable to all organizations that are assigned the cross-referenced Inventory Item.

You first add the customer item cross-reference records to the MTL_CI_XREFS_INTERFACE table. Then the Customer Item Cross-Reference Interface validates each record and moves the successfully validated items to the MTL_CUSTOMER_ITEM_XREFS table. When customer item cross-references are validated, the same rules are used as for the Customer Item Cross-References window to ensure that only valid cross-references are imported.

> **Note:** The Customer Item Open Interface must be run successfully before the Customer Item Cross-Reference Interface. This is to ensure that a customer item was defined before an attempt is made to cross-reference it with an Oracle Inventory item. The Customer Item Cross-Reference Interface errors out if an attempt is made to create a cross-reference to an invalid inventory item.

Workflow-Customer Item Open Interface and Customer Item Cross-Reference Interface

Before you use the Customer Item and Customer Item Cross-Reference Interfaces, you must write and run custom programs that extract customer item and customer item cross-reference information from your source system and insert it into the MTL_CI_INTERFACE and MTL_CI_XREFS_INTERFACE tables. After you load the customer items and customer item cross-references into these interface tables, you run the Customer Item and Customer Item Cross-Reference Interfaces to import the data. These interfaces assign defaults, validate data that you include, and then import the new customer items and customer item cross-references.

Interface Run-Time Options

You can access the Customer Item and Customer Item Cross-Reference Interfaces through the Reports, All menu in Oracle Inventory. (See: Importing Customer Items, *Oracle Inventory User's Guide* and Importing Customer Item Cross References, *Oracle Inventory User's Guide*.) Both Interfaces offer two options at run time: Abort on Error and Delete Successful Records.

Abort on Error

Valid values for this option are Yes and No. The default is No.

Yes: Both the Customer Item Open Interface and the Customer Item Cross-Reference Interface terminate processing if an error is encountered during validation of a record. No additional records are processed. The ERROR_CODE and ERROR_EXPLANATION columns in the MTL_CI_INTERFACE and MTL_CI_XREFS_INTERFACE tables are populated with the appropriate error code and error explanation for the record that caused the interface to error out. Records that were successfully validated are transferred to the MTL_CUSTOMER_ITEMS and MTL_CUSTOMER_ITEM_XREFS tables, respectively.

No: Processing of the records in the interface tables continues until the end of the table is reached. For all errors that are encountered during validation of records in the Customer Item Open Interface or Customer Item Cross-Reference Interface, the ERROR_CODE and the ERROR_EXPLANATION columns in the MTL_CI_INTERFACE and MTL_CI_XREFS_INTERFACE tables are populated with the appropriate error code and error description. Records that were successfully validated are transferred to the MTL_CUSTOMER_ITEMS and MTL_CUSTOMER_ITEM_XREFS tables, respectively.

Delete Successful Records

Valid values for this option are Yes and No. The default is Yes.

Yes: Successfully validated records in the Customer Item Open Interface are copied over to the MTL_CUSTOMER_ITEMS table and automatically deleted from the MTL_CI_INTERFACE table. Similarly, for the Customer Item Cross-Reference Interface, successfully validated records are copied to the MTL_CUSTOMER_ITEM_XREFS table

and automatically deleted from the MTL_CI_XREFS_INTERFACE table.

No: For successfully validated records, the Customer Item Open Interface and Customer Item Cross-Reference Interface populate the MTL_CUSTOMER_ITEMS and MTL_CUSTOMER_ITEM_XREFS tables without deleting records from the interface tables.

Customer Item Open Interface Table

Table Description

The Customer Item Open Interface table, MTL_CI_INTERFACE, includes all the columns in the Customer Items table, MTL_CUSTOMER_ITEMS. The columns are discussed after the table.

> **Note:** Information about columns that must be populated for audit trail and maintenance purposes can be found in the Table Administration and Audit Trail section. See Cycle Count Entries Interface.

For information about columns not discussed in the this manual, see Table and View Definitions, Electronic Technical Reference Manual (eTRM).

Field Name	Type	Required
PROCESS_FLAG	Varchar2(1)	x
PROCESS_MODE	Number	x
LAST_UPDATED_BY	Number	x
LAST_UPDATE_DATE	Date	x
LAST_UPDATE_LOGIN	Number	
CREATED_BY	Number	x
CREATION_DATE	Date	x
REQUEST_ID	Number	
PROGRAM_APPLICATION_ID	Number	

Field Name	Type	Required
PROGRAM_ID	Number	
PROGRAM_UPDATE_DATE	Date	
TRANSACTION_TYPE	Varchar2(6)	x
CUSTOMER_NAME	Varchar2(50)	Conditionally
CUSTOMER_NUMBER	Varchar2(30)	Conditionally
CUSTOMER_ID	Number	Conditionally
CUSTOMER_CATEGORY_CODE	Varchar2(30)	Conditionally
CUSTOMER_CATEGORY	Varchar2(80)	Conditionally
ADDRESS1	Varchar2(240)	Conditionally
ADDRESS2	Varchar2(240)	Conditionally
ADDRESS3	Varchar2(240)	Conditionally
ADDRESS4	Varchar2(240)	Conditionally
CITY	Varchar2(50)	Conditionally
STATE	Varchar2(50)	Conditionally
COUNTY	Varchar2(50)	Conditionally
COUNTRY	Varchar2(50)	Conditionally
POSTAL_CODE	Varchar2(30)	Conditionally
ADDRESS_ID	Number	Conditionally
CUSTOMER_ITEM_NUMBER	Varchar2(50)	x

Field Name	Type	Required
CUSTOMER_ITEM_DESC	Varchar2(240)	
ITEM_DEFINITION_LEVEL	Varchar2(1)	Conditionally
ITEM_DEFINITION_LEVEL_DESC	Varchar2(30)	Conditionally
MODEL_CUSTOMER_ITEM_NUMBER	Varchar2(50)	
MODEL_CUSTOMER_ITEM_ID	Number	
COMMODITY_CODE	Varchar2(30)	
COMMODITY_CODE_ID	Number	
MASTER_CONTAINER_SEGMENTn	Varchar2(40)	
MASTER_CONTAINER	Varchar2(2000)	
MASTER_CONTAINER_ITEM_ID	Number	
CONTAINER_ITEM_ORG_NAME	Varchar2(60)	
CONTAINER_ITEM_ORG_CODE	Varchar2(3)	
CONTAINER_ITEM_ORG_ID	Number	
DETAIL_CONTAINER_SEGMENTn	Varchar2(40)	
DETAIL_CONTAINER	Varchar2(2000)	
DETAIL_CONTAINER_ITEM_ID	Number	

Field Name	Type	Required
MIN_FILL_PERCENTAGE	Number	
DEP_PLAN_REQUIRED_FLAG	Varchar2(1)	
DEP_PLAN_PRIOR_BLD_FLAG	Varchar2(1)	
INACTIVE_FLAG	Varchar2(1)	x
ATTRIBUTE_CATEGORY	Varchar2(30)	
ATTRIBUTEn	Varchar2(150)	
DEMAND_TOLERANCE_POSITIVE	Number	
DEMAND_TOLERANCE_NEGATIVE	Number	
ERROR_CODE	Varchar2(9)	
ERROR_EXPLANATION	Varchar2(2000)	

Customer Item Open Interface - Defining a Unique Customer Item

You must define a unique record in each row of the MTL_CI_INTERFACE table. To create a unique record in the MTL_CI_INTERFACE table, you must define a customer item and the associated customer, category code, address, and item definition level for each record.

Customer Item

To create a unique customer item record in the MTL_CI_INTERFACE table, you must define a customer item number and customer item description in the CUSTOMER_ITEM_NUMBER and CUSTOMER_ITEM_DESC fields, respectively. The CUSTOMER_ITEM_NUMBER is a required field and is sufficient by itself for validation. However, Oracle recommends that you populate the CUSTOMER_ITEM_DESC field with accurate information to clearly identify customer items by their description.

Customer

You define a customer by populating the CUSTOMER_NAME, CUSTOMER_NUMBER, or CUSTOMER_ID field. Note that you must enter at least one of these fields for validation to occur. The information provided in these fields is validated against the Oracle table RA_CUSTOMERS. The interface errors out with the appropriate error code if the customer information cannot be validated. If more than one field is populated to identify a customer, then only the data in the highest priority field is used for validation according to the following rules of precedence:

- CUSTOMER_ID has priority over CUSTOMER_NUMBER.
- CUSTOMER_NUMBER has priority over CUSTOMER_NAME.

Address Category

Address category is a grouping of multiple customer ship-to addresses that have been defined in the RA_ADDRESSES table. This grouping is based on functional rules specific to your business and lets you select multiple customer addresses by specifying the address category. You can define the address category in the interface tables by populating either one of the CUSTOMER_CATEGORY_CODE or the CUSTOMER_CATEGORY field. However, these are conditionally required fields and may be null. Address category information is required if item definition level is set to 2 or item definition level description is set to address category. Any information that is entered in these fields is validated against the RA_ADDRESSES table.

If both fields are populated to define an address category, then only data in the highest priority field is used for validation against the RA_ADDRESSES table according to the following rule of precedence:

- CUSTOMER_CATEGORY_CODE has precedence over CUSTOMER_CATEGORY.

Customer Address

You can define the customer address information by entering either the detail customer address or the customer ADDRESS_ID. You must enter the detail customer address information, including the street address (ADDRESS1, ADDRESS2, ADDRESS3, ADDRESS4), CITY, STATE, COUNTY, COUNTRY and POSTAL_CODE, exactly as it is entered in the Oracle RA_ADDRESSES table, including any blank spaces, special characters, and capitalized alphabets. The customer address that you enter must match the information in the RA_ADDRESSES table exactly for successful validation.

Alternatively, you can enter the customer's ADDRESS_ID. This is also validated against the RA_ADDRESSES table, and detail customer address information is picked up from this table for successful validation.

Customer address information is required if the item definition level is set to 3 or the item definition level description is set to Address.

Customer Item Definition Level

A customer item can be defined at one of three levels: Customer level, address category level, or address level.

A customer item that is defined at the customer level is recognized across all addresses and address categories for that customer. However, if you ship an item to multiple customer ship-to sites that have been grouped as an address category, then you can define the customer item for that specific address category. You can define a customer item at the address level if you ship the item to only one ship-to site for a customer.

You must define the customer item definition level by populating either the ITEM_DEFINITION_LEVEL or the ITEM_DEFINITION_LEVEL_DESC column. Valid values for the ITEM_DEFINITION_LEVEL are 1, 2, and 3. The corresponding values for ITEM_DEFINITION_LEVEL_DESC are seeded in the MFG_LOOKUPS table and are customer, address category, and address, respectively.

If both fields are populated to identify the item definition level, then only data in the highest priority field is used for validation according to the following rule of precedence:

- ITEM_DEFINITION_LEVEL has higher priority than the ITEM_DEFINITION_LEVEL_DESC

Customer Item Open Interface - Other Fields

Transaction Type

TRANSACTION_TYPE is a required field. The interface errors out if a required field is missing or contains invalid data. Always set the TRANSACTION_TYPE to CREATE to create a new item in the MTL_CUSTOMER_ITEMS table. This is the only value that is supported by this interface.

Model Items

You can define a customer item as a model item that can be referenced by other customer items. To define a model customer item, you must first reference the customer item to a valid Oracle model item, which has the BOM Item Type attribute set to Model. (See: Bills of Material Attribute Group, *Oracle Inventory User's Guide*.) After a model customer item is defined successfully, you can reference other customer items to it.

The interface performs validation starting with the first record in MTL_CI_INTERFACE until it reaches the end of the table, or errors out if the Abort on Error option is set to Yes. Thus, the base model customer item must precede any customer items that reference it in the MTL_CI_INTERFACE table. The base model customer item must be validated successfully before other model customer items can be created that reference it; otherwise, the interface errors out.

You can define a model customer item by either specifying the

MODEL_CUSTOMER_ITEM_ID or the MODEL_CUSTOMER_ITEM_NUMBER. If both the fields are specified, then MODEL_CUSTOMER_ITEM_ID has precedence over MODEL_CUSTOMER_ITEM_NUMBER for validation. Any information in MODEL_CUSTOMER_ITEM_NUMBER is completely ignored in this case.

Commodity Codes

Commodity codes are used to group customer items in much the same way as the use of category codes to group inventory items. The business functionality and meaning of these codes are user-defined. For example, after the MTL_CUSTOMER_ITEMS table has been successfully populated, commodity codes can be used to query all customer items that belong to a specific commodity code.

Commodity codes are defined at the master organization level in Oracle Inventory and, thus, are applicable to all organizations that belong to the master organization. You must define the commodity code by specifying either the COMMODITY_CODE or the COMMODITY_CODE_ID for each customer item. If both fields are populated to define a commodity code, then only data in the highest priority field is used for validation according to the following rule of precedence:

- COMMODITY_CODE_ID has higher priority than the COMMODITY_CODE

Containers

A container item could be a pallet, box, bag, or any other inventory item that needs to be tracked between a customer and a supplier. Container items can be defined by setting the container item attribute to Yes. See: Physical Attribute Group, *Oracle Inventory User's Guide*.

In the Customer Item Open Interface, you set the default master or detail container for a customer item. A detail container is a subunit, or the inner container, of a larger outer unit, the master container. For example, a box can be a detail container for a customer item, while a pallet can be its master container.

Containers are defined at the master organization level in Oracle Inventory. You must specify the master organization for each master container in the Customer Item Open Interface. Similarly, you must specify the master container for each detail container. The interface errors out with the appropriate error code if no master organization is specified for a master container or if no master container is specified for a detail container.

A master container can be defined by populating either the MASTER_CONTAINER or the MASTER_CONTAINER_ITEM_ID field. Alternatively, you can use the MASTER_CONTAINER_SEGMENTn field to specify a multisegment container. If more than one field is populated to identify a master container, then only data in the highest priority field is used for validation according to the following rules of precedence:

- MASTER_CONTAINER_ITEM_ID has priority over MASTER_CONTAINER.

- MASTER_CONTAINER has priority over SEGMENTn values.

Similarly, a detail container can be defined by populating the DETAIL_CONTAINER, the DETAIL_CONTAINER_ITEM_ID, or the SEGMENTn values for the descriptive flexfield. The same rules of precedence apply as for preceding master container.

> **Warning:** The interface derives the segment values of each item by searching for the segment separator to indicate the end of one segment value and the start of the next segment value. Include the appropriate segment separator when populating the MASTER_CONTAINER or the DETAIL_CONTAINER field for a multisegment key flexfield.

> **Warning:** If you enter values for SEGMENTn columns, then ensure that the segment values you populate correspond to the key flexfield segments that you defined for your items. The interface assumes that you are using segments in sequential order beginning with SEGMENT1.

You can also specify the MIN_FILL_PERCENTAGE attribute when you define the master container. The Minimum Fill Percentage item attribute defines the minimum percentage of the master, or outer container, that should be filled by the detail, or inner container, before the master container can be shipped. See: Physical Attribute Group, *Oracle Inventory User's Guide*.

Departure Planning Flags

The DEP_PLAN_REQUIRED_FLAG and DEP_PLAN_PRIOR_BLD_FLAG fields can have the values of 1 for Yes and 2 for No.

The DEP_PLAN_REQUIRED_FLAG is used to signal Oracle Shipping to perform Departure Planning for this item. The DEP_PLAN_PRIOR_BLD_FLAG is used to indicate that Departure Planning is required before building this item. If the DEP_PLAN_PRIOR_BLD_FLAG is Yes, then the DEP_PLAN_REQUIRED_FLAG must also be set to Yes.

Inactive Flag

INACTIVE_FLAG is a required field and can be set to 1 for Yes or 2 for No. You can set the INACTIVE_FLAG to Yes to deactivate a customer item in Oracle Inventory. The customer item information is still carried over from MTL_CI_INTERFACE to the MTL_CUSTOMER_ITEMS table if the record is successfully validated. However, the Customer Item is considered as status Inactive in Oracle Inventory.

Descriptive Flex- SEGMENTn

The ATTRIBUTE_CATEGORY and ATTRIBUTE1 - ATTRIBUTE15 columns are used for the descriptive flexfield information. Note that the interface does not perform any

validation on the SEGMENTn fields even though you may have defined a valid value set for the descriptive flexfield.

Demand Tolerance Range

The DEMAND_TOLERANCE_POSITIVE and DEMAND_TOLERANCE_NEGATIVE fields are used to define the percentage range within which the order quantities for a customer item are acceptable. This range is based on a customer's last order for the same item. No range validation is performed for the first order of an item because no history information exists in the demand tables.

If a customer order falls outside the range that is defined by these fields then the demand processor raises an exception to that order. This feature is designed to flag any Order Management or EDI transfer errors and to draw your attention toward substantial changes in order volume activity from a customer.

Error Codes

If any errors are found during validation, then the error codes and the corresponding error description are written to the ERROR_CODE and the ERROR_EXPLANATION columns, respectively. Note that the interface overwrites any data already in these fields.

Record Status

The PROCESS_FLAG and PROCESS_MODE columns report the status of the record after the import and validation process is complete. Data already in these columns is ignored and overwritten, if necessary. Note that these are required columns and should be populated with 1.

Customer Item Cross-Reference Interface Table

Table Description

The Customer Item Cross-Reference Interface table, MTL_CI_XREFS_INTERFACE, contains all the columns in the Customer Item Cross-Reference table, MTL_CUSTOMER_ITEM_XREFS.

Many columns in the Customer Item Cross-Reference Interface table are similar to columns in the Customer Item Open Interface table. Columns that identify the customer, customer category, address, and item definition level are subject to the same rules and definitions as those described for the Customer Item Open Interface table. You can also refer to the previous section for explanations of descriptive flexfields, PROCESS_FLAG and PROCESS_MODE, ERROR_CODE and ERROR_EXPLANATION, INACTIVE FLAG, and TRANSACTION_TYPE columns. See Customer Item Open Interface Table.

Note: Information about columns that must be populated to create an audit trail and for maintenance purposes can be found in the Table Administration and Audit Trail section. See: Cycle Count Entries Interface.

Note: For information about columns not discussed in this manual, see Table and View Definitions, Electronic Technical Reference Manual (eTRM).

This section provides an explanation of fields that are used to define the inventory item, master organization, and rank in the MTL_CI_XREFS_INTERFACE table.

Field Name	Type	Required	Derived	Optional
PROCESS_FLAG	Varchar2(1)	x		
PROCESS_MODE	Number	x		
LAST_UPDATE_DATE	Date	x		
LAST_UPDATED_BY	Number(15)	x		
CREATION_DATE	Date	x		
CREATED_BY	Number(15)	x		
LAST_UPDATE_LOGIN	Number(15)			
REQUEST_ID	Number(15)			
PROGRAM_APPLICATION_ID	Number(15)			
PROGRAM_ID	Number(15)			
PROGRAM_UPDATE_DATE	Date			

Field Name	Type	Required	Derived	Optional
TRANSACTION_TYPE	Varchar2(6)	x		
CUSTOMER_NAME	Varchar2(50)	Conditionally		
CUSTOMER_NUMBER	Varchar2(30)	Conditionally		
CUSTOMER_ID	Number	Conditionally		
CUSTOMER_CATEGORY_CODE	Varchar2(30)	Conditionally		
CUSTOMER_CATEGORY	Varchar2(80)	Conditionally		
ADDRESS1	Varchar2(240)	Conditionally		
ADDRESS2	Varchar2(240)	Conditionally		
ADDRESS3	Varchar2(240)	Conditionally		
ADDRESS4	Varchar2(240)	Conditionally		
CITY	Varchar2(50)	Conditionally		
STATE	Varchar2(50)	Conditionally		
COUNTY	Varchar2(50)	Conditionally		
COUNTRY	Varchar2(50)	Conditionally		
POSTAL_CODE	Varchar2(30)	Conditionally		
ADDRESS_ID	Number	Conditionally		
CUSTOMER_ITEM_NUMBER	Varchar2(50)	x		

Field Name	Type	Required	Derived	Optional
CUSTOMER_ITEM_ID	Number			
ITEM_DEFINITION_LEVEL_DESC	Varchar2(30)	Conditionally		
ITEM_DEFINITION_LEVEL	Varchar2(1)	Conditionally		
INVENTORY_ITEM_SEGMENTn	Varchar2(40)	Conditionally		
INVENTORY_ITEM	Varchar2(2000)	Conditionally		
INVENTORY_ITEM_ID	Number	Conditionally		
MASTER_ORGANIZATION_NAME	Varchar2(60)	Conditionally		
MASTER_ORGANIZATION_CODE	Varchar2(3)	Conditionally		
MASTER_ORGANIZATION_ID	Number	Conditionally		
PREFERENCE_NUMBER	Number	x		
INACTIVE_FLAG	Varchar2(1)	x		
ATTRIBUTE_CATEGORY	Varchar2(30)			
ATTRIBUTEn	Varchar2(150)			
ERROR_CODE	Varchar2(9)			

Field Name	Type	Required	Derived	Optional
ERROR_EXPLANATION	Varchar2(2000)			

Inventory Item

You must define a valid inventory item for each customer item to define a successful cross-reference relationship. The inventory item must be a valid item in the master organization for which the cross-reference is being defined. The interface errors out with the appropriate error code if an invalid inventory item is specified.

You can define an inventory item by specifying either the INVENTORY_ITEM or the INVENTORY_ITEM_ID. Alternatively, you can define an inventory item by specifying the SEGMENTn values of the item key flexfield for a multisegment item. The inventory item is validated against the MTL_SYSTEM_ITEMS table for the specified master organization.

If more than one field is populated to identify an inventory item, then only data in the highest priority field is used for validation according to the following rules of precedence:

- INVENTORY_ITEM_ID has priority over INVENTORY_ITEM.
- INVENTORY_ITEM has priority over SEGMENTn values.

Master Organization

For each inventory item in the Customer Item Cross-Reference table, you must also specify the master organization for which the cross-reference is being defined. The interface errors out with the appropriate error code if an invalid master organization is specified.

You can define a master organization by specifying either the MASTER_ORGANIZATION_ID or the MASTER_ORGANIZATION_CODE of the organization. If both fields are specified, then MASTER_ORGANIZATION_ID has precedence over MASTER_ORGANIZATION_CODE.

Rank

You can define multiple references between a customer item and several inventory items. They can be used to define alternate, or substitute, inventory items for a customer item. In this case, a preference ranking is established to determine the item that must be processed in Oracle Inventory when a customer item is demanded.

You must specify the rank of the cross-referenced relationship for each cross-reference that you define. The top-ranked inventory item is processed before a lower-ranked item in the supplying organization. Thus, an inventory item with a rank of 1 is processed

before another item with a rank of 2 that references the same customer item. This is a required field.

15

Oracle Warehouse Management Open Interfaces and APIs

This chapter covers the following topics:

- Compliance Label Application Program Interface
- Oracle Warehouse Management Device Integration API
- Locator Maintenance API
- Container API
- Oracle Warehouse Management Installation API
- Radio Frequency Identifier APIs
- Close Truck and Ship Confirm API
- Electronic Product Code API
- Crossdock Customization API
- Rules Engine Custom API

Compliance Label Application Program Interface

You can use this application program interface (API) to print labels within Oracle Warehouse Management. It is the same API that is used to create a manual print request from the Oracle Warehouse Management mobile user interface. The package that includes this API is INV_LABEL.

Use the PRINT_LABEL_MANUAL_WRAP procedure to print labels using the INV_LABEL package. Using the PRINT_LABEL_MANUAL_WRAP procedure, the user can pass the API Label Type to be printed, as well as the particular data to include on the label. For the label printing to be successful, the label printing setup must be complete within Oracle Warehouse Management, with the exception of the Business Flow assignment. Because the label type is being passed as a parameter of this API, you do not need to derive the label type from a Business Flow assignment using this API.

For instance, if the requirement is to print an LPN Contents label type from somewhere other than where Oracle Warehouse Management supports such printing, then you can call this API by passing the label type parameter to refer to LPN Contents, and passing the identifier of the LPN for which the label should be printed.

Functional Overview

This API invokes the Oracle Warehouse Management Rules Engine to determine the proper label format to be printed for the selected label type. During Oracle Warehouse Management setup, you also select the printer you want to use.

INV_LABEL. PRINT_LABEL_MANUAL_WRAP

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
x_label_status	OUT	Varchar2			
p_business_flow_code	IN	Number			
p_label_type	IN	Number			
p_organization_id	IN	Number			
p_inventory_item_id	IN	Number			
p_lot_number	IN	Varchar2			
p_fm_serial_number	IN	Varchar2			
p_to_serial_number	IN	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
p_lpn_id	IN	Number			
p_subinventory_code	IN	Varchar2			
p_locator_id	IN	Number			
p_delivery_id	IN	Number			
p_quantity	IN	Number			
p_uom	IN	Varchar2			
p_no_of_copies	IN	Number			
p_format_id	IN	Number			
p_printer_name	IN	Varchar2			

x_return_status

Returns the status of the API request.

x_msg_count

Returns the count of error messages that are included in x_msg_data.

x_msg_data

Returns the content of error messages.

x_label_status

Returns the status of the print request. This value currently always returns a success value. With later integration to synchronous print request, it will return status from synchronous print call.

p_business_flow_code

Indicates the business flow from which the label is being printed. This field is not used for manual print requests.

p_label_type

Indicates the label type to print. This is a numeric lookup value from the list that is seeded in the table MFG_LOOKUPS with a LOOKUP_TYPE of WMS_LABEL_TYPE. Valid values are:

Lookup Code	Label Type
1	Material
2	Serial
3	LPN
4	LPN Content
5	LPN Summary
6	Location
7	Shipping
8	Shipping Contents

p_organization_id

Indicates the organization ID for the label information being printed. It is used when printing all label types.

p_inventory_item_id

Indicates the inventory item ID for the material being printed. Use this parameter when printing a label of type Material or Serial only.

p_lot_number

Indicates the lot for the material being printed. Use this parameter when printing the lot for a label of type Material or Serial only.

p_fm_serial_number

Indicates the starting serial number for the material being printed. Use this parameter to print the serial numbers for a label of type Serial only.

p_to_serial_number

Indicates the ending serial number for the material being printed. Use this parameter to print the serial numbers for a label of type Serial only.

p_lpn_id

Indicates the License plate number ID for the license plate number that is being printed. Use this parameter for label types LPN, LPN Contents, LPN Summary, and Shipping Contents.

p_subinventory_code

Indicates the Subinventory Code being printed. Use this parameter for label type Location only.

p_locator_id

Indicates the identifier of the locator being printed. Use this parameter for label type Location only.

p_delivery_id

Indicates the identifier of the delivery being printed. Use this parameter for label types Shipping and Shipping Contents only.

p_quantity

Indicates the quantity of the material being printed. Use this parameter for label type Material only.

p_uom

Indicates the unit of measure of the material being printed. Use this parameter for label type Material only.

p_no_of_copies

Indicates the number of copies of the label being printed. Use this parameter for all label types.

p_format_id

Indicates the label format id to be used for the label to be printed. If provided, this specific label format will be used to generate labels. Otherwise, the label format will be decided with WMS Rules Engine or default label format in a Inventory only application. Default value is NULL.

p_printer_name

Indicates the printer name where the label needs to be printed. If it is provided, this specific printer will be used to print a label. Otherwise, the printer will be decided with printer setup. Default value is NULL.

Detailed Discussion of Parameters Required for Each Label Type

This section describes the parameters that are used for each label type.

Material Label Type

This label type requires the parameters p_organization_id, p_inventory_item_id, p_quantity, and p_uom. The parameter p_lot_number can be supplied if the item being printed is under lot control.

Serial Label Type

This label type requires the parameters p_organization_id, p_inventory_item_id, p_quantity, p_uom, and p_fm_serial_number. The parameter p_lot_number can be supplied if the item being printed is under lot control. The parameter p_to_serial_number can be supplied if a range of serials needs to be printed.

LPN Label Type

This label type requires the parameter p_lpn_id.

LPN Contents Label Type

This label type requires the parameter p_lpn_id.

LPN Summary Label Type

This label type requires the parameter p_lpn_id.

Location Label Type

This label requires the parameter p_organization_id. To print location labels for a particular subinventory, pass the parameter p_subinventory_code. To print labels for all subinventories within the organization, pass p_subinventory_code as NULL. To exclude subinventory information from the label, pass p_subinventory_code as -1. To print location labels for a particular locator, pass the parameter p_locator_id. To print labels for all locators within the subinventories that are being printed, pass p_locator_id as NULL. To exclude locator information from the label, pass p_locator_id as -1.

Shipping Label Type

This label type requires the parameter p_deliver_id.

Shipping Contents Label Type

To print this label type, you must pass either the parameter p_delivery_id or p_lpn_id. If you pass p_delivery_id, then labels are printed for each delivery detail that is associated with that delivery. If you pass p_lpn_id, then labels are printed for each delivery detail that is associated with only that LPN.

Oracle Warehouse Management Device Integration API

Oracle Warehouse Management needs to support integration with various types of devices such as carousels, conveyers, and scales that are typically used in a warehouse. This requires a flexible architecture that facilitates integration with different types of devices over a variety of interface modes, without being tied to any specific vendor or method.

Integration requirements:

- Ability to integrate with a variety of device types including carousels, conveyers, pick-to-light systems, voice-picking systems, and scales.
- Ability to initiate a request to a device from a radio frequency device or from an Oracle Warehouse Management business operation module.
- Ability to communicate with devices or device-integration third-party software through a variety of configurable methods like XML/DTD, PL/SQL API or database tables.
- Ability to allow association of specific device classes to business events. (Example: Map carousels and conveyers for the pick confirm business event.)
- Ability to select a device based on the attributes of the business event instance in which the request is initiated. (Example: Select Carousel A when zone is RECV.)
- Ability to receive and process responses from a device.

This API is provided as a stub that is extended by device integration vendors. This API has visibility to the temporary table WMS_DEVICE_REQUESTS. The OUT parameters can be used to send back a status or message from the device to Oracle Warehouse Management. To get details of the request, WMS_DEVICE_REQUESTS must be joined with other tables.

Parameter	Usage	Type	Required	Derived	Optional
p_request_id	IN	Number			
p_device_id	IN	Number			
p_resubmit_flag	IN	Varchar2			
x_status_code	OUT	Varchar2			
x_status_msg	OUT	Varchar2			
x_device_status	OUT	Varchar2			

p_request_id

This unique request identifier in wms_device_requests is generated from the sequence WMS_DEVICE_REQUESTS_S.

p_device_id

This unique device identifier if the device to be invoked from wms_devices_tl.

p_resubmit_flag

Flag to indicate whether the API is invoked as part of resubmitting a request.

x_status_code

Status of request: Error or Success.

x_status_msg

Message about the status of the request.

x_device_status

Any information message about the device.

Device Confirmation API

The Device Confirmation API (WMS_DEVICE_CONFIRMATION) is used for device confirmation. It takes all records from the Wms_Device_Requests (WDR) temporary table for DEVICE_CONFIRMATION and transfers them to the Wms_Device_Requests_Hist (WDRH) table. The following table provides the specifications for this API:

Package Name: WMS_DEVICE_CONFIRMATION

Procedure Name: DEVICE_CONFIRMATION

WMS_DEVICE_CONFIRMATION. DEVICE_CONFIRMATION

Parameter	Usage	Type	Required	Derived	Optional
p_request_id	IN	Number			
x_return_status	OUT	Varchar2			
x_msg_data	OUT	Varchar2			
x_successful_row_cnt	OUT	Number			

p_request_id

Optional parameter that identifies a record in WMS_DEVICE_REQUESTS_HIST. It is used internally when a failed record is resubmitted from the Device History form.

x_return_status

Standard Oracle API output parameter.

x_msg_data

Standard Oracle API output parameter.

x_successful_row_cnt

Number of successful rows after processing.

Locator Maintenance API

Oracle Warehouse Management maintains available weight capacity, available volume capacity, and available units capacity of a locator in real time. This is essential for an efficient putaway process. During any material transaction, the available weight capacity, available volume capacity, and available units capacity of the transacted locator are updated suitably based on the total weight and total weight of the transacted item and maximum weight, maximum volume, and maximum units allowed for the locator. Therefore, you must define a weight unit of measure, maximum weight, volume unit of measure, maximum volume, and maximum units for all locators that are used in Oracle Warehouse Management.

The X, Y, and Z coordinates of a locator indicate its physical position in a warehouse. This information is used in task dispatching an Oracle Warehouse Management functionality that intelligently and efficiently dispatches tasks to warehouse personnel in real time. During dispatching, the system considers, among other things, the current location of warehouse personnel. The location is the physical position of the last locator that was worked on. Therefore, the physical location of a locator is essential for efficient task dispatching.

For existing Oracle Applications customers who are upgrading to Oracle Warehouse Management, Oracle recommends that you run the set of check scripts that are provided with the product, and take necessary corrective action taken before using Oracle Warehouse Management. These scripts identify data that was in existence before Oracle Warehouse Management that might need to be enhanced to better utilize the capabilities of Oracle Warehouse Management. One such script is the Locator check script. It lists locators with one of the following not defined:

- Weight unit of measure
- Maximum weight
- Volume unit of measure
- Maximum volume
- Maximum units
- X coordinate
- Y coordinate
- Z coordinate
- Dimension unit of measure
- Length

- Width
- Height

You can use the existing locators maintenance form to add the missing attributes to the definition of the listed locators. Data entry can be time consuming if the locators list is large. A faster approach is to write a script in an appropriate language that repeatedly calls APIs that perform locator maintenance by passing the required data.

In Oracle Inventory, items can be restricted to certain locators, but with Oracle Warehouse Management the same functionality is used to assign items to locators to create a soft assignment that can be used by the rules engine. In Oracle Inventory, you cannot delete locators. To reduce the volume of data and improve performance, delete locators that are obsolete. Before deleting locators, confirm that the locator you plan to delete does not exist in any core Inventory table.

Locator Maintenance APIs handle the preceding requirements. They constitute the following:

- Create Locator API to create a new locator (CREATE_LOCATOR).
- Update Locator API to update an existing locator (UPDATE_LOCATOR).
- Create Locator Item Tie API to assign an item to a locator. (CREATE_LOC_ITEM_TIE)
- Delete Locator API to delete an existing locator (DELETE_LOCATOR).

The APIs are part of PL/SQL package INV_LOC_WMS_PUB. This is defined in $INV_TOP/patch/115/sql/INVLOCPS.pls. A description of each API follows.

Create Locator API

For given concatenated locator segments and organizations, this API creates a new locator in the organization and returns the locator identifier. If a locator already exists with the same concatenated segments, then the API returns the locator identifier.

INV_LOC_WMS_PUB.CREATE_LOCATOR

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			

Parameter	Usage	Type	Required	Derived	Optional
x_msg_data	OUT	Varchar2			
x_inventory_location_id	OUT	Number			
x_locator_exists	OUT	Varchar2			
p_organization_id	IN	Number			
p_organization_code	IN	Varchar2			
p_concatenated_segments	IN	Varchar2			
p_description	IN	Varchar2			
p_inventory_location_type	IN	Number			
p_picking_order	IN	Number			
p_location_maximum_units	IN	Number			
p_subinventory_code	IN	Varchar2			
p_location_weight_uom_code	IN	Varchar2			
p_max_weight	IN	Number			
p_volume_uom_code	IN	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
p_max_cubic_area	IN	Number			
p_x_coordinate	IN	Number			
p_y_coordinate	IN	Number			
p_z_coordinate	IN	Number			
p_physical_location_id	IN	Number			
p_pick_uom_code	IN	Varchar2			
p_dimension_uom_code	IN	Varchar2			
p_length	IN	Number			
p_width	IN	Number			
p_height	IN	Number			
p_status_id	IN	Number			

x_return_status

Return status indicating success (S), error (E), unexpected error (U).

x_msg_count

Number of messages in the message list.

x_msg_data

If the number of messages in message list is 1, then this parameter contains message text.

x_inventory_location_id

Identifier of a newly created locator or existing locator.

x_locator_exists

A value of Y indicates that the locator exists for given input.

A value of N indicates that the locator was created for a given input.

p_organization_id

Identifier of the organization in which you plan to create the locator.

p_organization_code

Code of the organization in which you plan to create the locator. You must pass either the p_organization_id or p_organization_code.

p_concatenated_segments

Concatenated segment string with the locator separator that you plan to create. Example: A.1.1

p_description

Locator description.

p_inventory_location_type

Type of locator. The valid values are dock door (1), staging lane (2), and storage locator (3).

p_picking_order

Number that identifies the relative position of the locator for travel optimization during task dispatching. It has a higher precedence over the x, y, and z coordinates.

p_location_maximum_units

Maximum units that the locator can hold.

p_subinventory_code

Subinventory to which the locator belongs.

p_location_weight_uom_code

Unit of measure of the maximum weight capacity of the locator.

p_max_weight

Maximum weight that the locator can hold.

p_volume_uom_code

Unit of measure of the maximum volume capacity of the locator.

p_max_cubic_area

Maximum volume capacity of the locator.

p_x_coordinate

Position of the locator in space. Used for travel optimization during task dispatching.

p_y_coordinate

Position of the locator in space. Used for travel optimization during task dispatching.

p_z_coordinate

Position of the locator in space. Used for travel optimization during task dispatching.

p_physical_location_id

Locators that are the same physically have the same inventory_location_id in this column.

p_pick_uom_code

Unit of measure in which the material is picked from the locator.

p_dimension_uom_code

Unit of measure in which the locator dimensions are expressed.

p_length

Length of the locator.

p_width

Width of the locator.

p_height

Height of the locator.

Update Locator API

This API updates an existing locator with the information that is provided as input parameters. If the default value is passed, then the corresponding locator column retains its original value. You can achieve this result by not passing that parameter during the API call.

INV_LOC_WMS_PUB.CREATE_LOCATOR .UPDATE_LOCATOR

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_organization_id	IN	Number			
p_organization_code	IN	Varchar2			
p_inventory_location_id	IN	Number			
p_concatenated_segments	IN	Varchar2			
p_description	IN	Varchar2			
p_disabled_date	IN	Date			
p_inventory_location_type	IN	Number			
p_picking_order	IN	Number			

Parameter	Usage	Type	Required	Derived	Optional
p_location_maximum_units	IN	Number			
p_location_weight_uom_code	IN	Varchar2			
p_max_weight	IN	Number			
p_volume_uom_code	IN	Varchar2			
p_max_cubic_area	IN	Number			
p_x_coordinate	IN	Number			
p_y_coordinate	IN	Number			
p_z_coordinate	IN	Number			
p_physical_location_id	IN	Number			
p_pick_uom_code	IN	Varchar2			
p_dimension_uom_code	IN	Varchar2			
p_length	IN	Number			
p_width	IN	Number			
p_height	IN	Number			
p_status_id	IN	Number			

x_return_status

Return status indicating success (S), error (E), unexpected error (U).

x_msg_count

Number of messages in the message list.

x_msg_data

If the number of messages in the message list is 1, then this parameter contains message text.

p_organization_id

Identifier of the organization in which you plan to create the locator.

p_organization_code

Code of the organization in which you plan to create the locator. You must pass either p_organization_id or p_organization_code.

p_inventory_location_id

Identifier of the locator that you want to update.

p_concatenated_segments

Concatenated segment string with separator of the locator to be created. Example: A.1.1. You must pass either p_inventory_location_id or p_concatenated_segments.

p_description

Locator description

p_inventory_location_type

Type of locator. The valid values are dock door (1), staging lane (2), and storage locator (3).

p_picking_order

Number that identifies relative position of locator for travel optimization during task dispatching. It has a higher precedence over x, y, and z coordinates.

p_location_maximum_units

Maximum units that the locator can hold.

p_subinventory_code

Subinventory to which the locator belongs.

p_location_weight_uom_code

Unit of measure of the maximum weight capacity of the locator.

p_max_weight

The maximum weight that the locator can hold.

p_volume_uom_code

Unit of measure of the maximum volume capacity of the locator.

p_max_cubic_area

Maximum volume capacity of the locator.

p_x_coordinate

Position of the locator in space. Used for travel optimization during task dispatching.

p_y_coordinate

Position of the locator in space. Used for travel optimization during task dispatching.

p_z_coordinate

Position of the locator in space. Used for travel optimization during task dispatching.

p_physical_location_id

Locators that are the same physically have the same inventory_location_id in this column.

p_pick_uom_code

Unit of measure in which material is picked from the locator.

p_dimension_uom_code

Unit of measure in which the locator dimensions are expressed.

p_length

Length of the locator.

p_width

Width of the locator.

p_height

Height of the locator.

p_status_id

Material status that must be associated with the locator.

Create Locator Item Tie API

For a given set of organizations, subinventories, items, and locators, this API ties the given item to the given locator.

INV_LOC_WMS_PUB.CREATE_LOCATOR.CREATE_LOC_ITEM_TIE

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_inventory_item_id	IN	Number			
p_item	IN	Varchar2			
p_organization_id	IN	Number			
p_organization_code	IN	Varchar2			
p_subinventory_code	IN	Varchar2			
p_inventory_location_id	IN	Number			

Parameter	Usage	Type	Required	Derived	Optional
p_locator	IN	Varchar2			
p_status_id	IN	Number			

x_return_status

Return status indicating success (S), error (E), unexpected error (U).

x_msg_count

Number of messages in the message list.

x_msg_data

If the number of messages in the message list is 1, then this column contains the message text.

p_inventory_item_id

Unique identifier of the item.

p_item

Concatenated segment string with separator of the item. You must pass either p_inventory_item_id or p_item.

p_organization_id

Unique identifier of the organization.

p_organization_code

Code of the organization in which you plan to update the locator. You must pass either p_organization_id or p_organization_code.

p_subinventory_code

The subinventory to which the tied locator belongs.

p_inventory_location_id

Identifier of the locator to be attached to the specified subinventory.

p_locator

Concatenated segment string with the separator of the locator that you want to update. Example: A.1.1. You must pass either p_inventory_location_id or p_concatenated_segments.

p_status_id

Identifier of the locator material status.

Delete Locator API

This API deletes an existing locator. The locator is deleted after the system ensures that the locator is obsolete and does not exist in any core Inventory tables.

INV_LOC_WMS_PUB. .DELETE_LOCATOR

Parameter	Usage	Type	Required	Derived	Optional
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_inventory_location_id	IN	Number			
p_concatenated_segments	IN	Varchar2			
p_organization_id	IN	Number			
p_organization_code	IN	Varchar2			
p_validation_req_flag	IN	Varchar2			

x_return_status

Return status indicating success (S), error (E), unexpected error (U).

x_msg_count

Number of messages in the message list.

x_msg_data

If the number of messages in message list is 1, then this column contains message text.

p_inventory_location_id

Identifier of the locator that you plan to delete.

p_concatenated_segments

Concatenated segment string with separator of the locator that you plan to delete. Example: A.1.1. You must pass either p_inventory_location_id or p_concatenated_segments.

p_organization_id

Identifier of the organization that contains the locator that you plan to delete.

p_organization_code

Code of the organization in which locator is to be deleted. You must pass either p_organization_id or p_organziation_code.

p_validation_req_flag

Flag which determines whether validation is required. If the column value is N, then the locator is deleted without any further validation on its existence in the core Inventory tables. If the column value is Y, then the locator is deleted only if does not exist in the core Inventory tables.

Container API

Container APIs are used to generate LPNs, pack containers, pre-pack containers, and unpack containers, and are used for validations.

Generate LPN API

This API generates LPN numbers, either individually or in a sequence.

WMS_Container_PUB.GENERATE_LPN

Parameter	Usage	Type	Required	Derived	Optional
p_api_version	IN	Number			
p_init_msg_list	IN	Varchar2			X
p_commit	IN	Varchar2			X
p_validation_level	IN	Number			X
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			
p_organization_id	IN	Number	X		
p_container_item_id	IN	Number			
p_revision	IN	Varchar2			
p_lot_number	IN	Varchar2			
p_from_serial_number	IN	Varchar2			
p_to_serial_number	IN	Varchar2			
p_subinventory	IN	Varchar2			
p_locator_id	IN	Number			

Parameter	Usage	Type	Required	Derived	Optional
p_lpn_prefix	IN	Varchar2			
p_lpn_suffix	IN	Varchar2			
p_starting_num	IN	Number			
p_quantity	IN	Number			
p_source	IN	Number			
p_cost_group_id	IN	Number			
p_source_type_id	IN	Number			
p_source_header_id	IN	Number			
p_source_name	IN	Varchar2			
p_source_line_id	IN	Number			
p_source_line_detail_id	IN	Number			
p_lpn_id_out	OUT	Number			
p_lpn_out	OUT	Varchar2			
p_process_id	OUT	Number			

p_api_version

API version number.

p_init_msg_list

Valid values: FND_API.G_FALSE or FND_API.G_TRUE. If you set this value to

FND_API.G_TRUE, then the API initializes the error message list. If you set this value to FND_API.G_FALSE, then the API does not initialize the error message list.

p_commit

This value determines whether to commit the changes to the database.

p_validation_level

This value determines whether to perform a full validation or no validation. Valid values are FND_API.G_VALID_LEVEL_FULL (default) and FND_API.G_VALID_LEVEL_NONE.

p_organization_id

Unique identifier of the organization ID.

p_container_item_id

Unique identifier of the container item ID.

p_revision

Revision number.

p_lot_number

Lot number.

p_from_serial_number

Starting serial number.

p_to_serial_number

Ending serial number.

p_subinventory

Subinventory name.

p_locator_id

Locator identifier.

p_lpn_prefix

Prefix value of an LPN.

p_lpn_suffix

Suffix value of an LPN.

p_starting_num

Starting number of an LPN.

p_quantity

Number of LPNs to be generated.

p_source

LPN Context. 1. INV, 2. WIP, or 2. REC, and so on.

p_cost_group_id

Cost group identifier.

p_source_type_id

Type identifier for the source transaction.

p_source_header_id

Header identifier for the source transaction.

p_source_name

Name of the source transaction.

p_source_line_id

Line identifier for the source transaction.

p_source_line_detail_id

Line detail identifier for the source transaction.

x_return_status

If Generate_LPN API succeeds, then the value is fnd_api.g_ret_sts_success. If an expected error occurs, then the value is fnd_api.g_ret_sts_error. If an unexpected error occurs, then the value is fnd_api.g_ret_sts_unexp_error.

x_msg_count

If one or more errors occur, then this value represents the number of error messages in

the buffer.

x_msg_data

If only one error occurs, then this column contains the error message. (See fnd_api package for more details about the preceding output parameters.)

p_lpn_id_out

Writes the generated LPN ID if only one LPN is requested to be generated.

p_lpn_out

Writes the generated license plate number if only one LPN is requested to be generated.

p_process_id

Process identifier that identifies the LPNs that were generated in table WMS_LPN_PROCESS_TEMP.

Create LPN API

This API associates an LPN with a specific instance of a container in which the LPN already exists. The API creates an associated LPN ID and stores the record in the WMS_LICENSE_PLATE_NUMBERS table.

WMS_CONTAINER_PUB.CREATE_LPN

Parameter	Usage	Type	Required	Derived	Optional
p_api_version	IN	Number			
p_init_msg_list	IN	Varchar2			
p_commit	IN	Varchar2			
p_validation_level	IN	Number			
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			

Parameter	Usage	Type	Required	Derived	Optional
x_msg_data	OUT	Varchar2			
p_lpn	IN	Varchar2			
p_organization_id	IN	Number			
p_container_item_id	IN	Number			
p_lot_number	IN	Varchar2			
p_revision	IN	Varchar2			
p_serial_number	IN	Varchar2			
p_subinventory	IN	Varchar2			
p_locator_id	IN	Number			
p_source	IN	Number			
p_cost_group_id	IN	Number			
p_parent_lpn_id	IN	Number			
p_source_type_id	IN	Number			
p_source_header_id	IN	Number			
p_source_name	IN	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
p_source_line_id	IN	Number			
p_source_line_detail_id	IN	Number			
x_lpn_id	OUT	Number			

p_api_version

API version number.

p_init_msg_list

Valid values: FND_API.G_FALSE or FND_API.G_TRUE. If you set this column to FND_API.G_TRUE, then the API initializes the error message list. If you set this column to FND_API.G_FALSE, then the API does not initialize the error message list.

p_commit

The value in this column determines whether to commit the changes to the database.

p_validation_level

The value in this column determines if the API performs full validation or no validation. The column default value is FND_API.G_VALID_LEVEL_FULL. FND_API.G_VALID_LEVEL_NONE is the other value.

p_lpn_id

Unique identifier of the license plate number.

p_container_item_id

Unique identifier of the container item.

p_lot_number

Lot number.

p_revision

Revision number.

p_serial_number

Serial number.

p_organization_id

Unique identifier of the organization.

p_subinventory

Subinventory name.

p_locator_id

Unique identifier of the locator.

p_cost_group_id

Unique identifier of the cost group.

p_source_type_id

Type identifier of the source transaction.

p_source_header_id

Header identifier of the source transaction.

p_source_name

Name of the source transaction.

p_source_line_id

Line identifier of the source transaction.

p_source_line_detail_id

Line detail identifier for the source transaction.

x_return_status

If the Create_LPN API succeeds, then the value is fnd_api.g_ret_sts_success. If an expected error occurs, then the value is fnd_api.g_ret_sts_error. If an unexpected error occurs, then the value is fnd_api.g_ret_sts_unexp_error.

x_msg_count

This column shows the number of error messages in the buffer, if one or more errors

occur.

x_msg_data

This column shows the error message, if only one error occurs.

x_lpn_id

The license plate number identifier for the new LPN record entry.

Modify LPN API

This API is used to update the attributes of a specific container instance (LPN). Fields that can be modified include:

- All fields related to gross weight and content volume
- Status_id, lpn_context, sealed_status
- Org, sub, and loc information
- All of the extra attribute-related columns for future usage

MODIFY_LPN

Parameter	Usage	Type	Required	Derived	Optional
p_api_version	IN	Number			
p_init_msg_list	IN	Varchar2			
p_commit	IN	Varchar2			
p_validation_level	IN	Number			
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			

p_lpn	IN	Varchar2
p_source_type_id	IN	Number
p_source_header_id	IN	Number
p_source_name	IN	Varchar2
p_source_line_id	IN	Number
p_source_line_		
detail_id	IN	Number

p_api_version

API version number.

p_init_msg_list

Valid values: FND_API.G_FALSE or FND_API.G_TRUE. If you set this parameter to FND_API.G_TRUE, then the API initializes the error message list. If you set this parameter to FND_API.G_FALSE, then the API does not initialize the error message list.

p_validation_level

This value determines whether to perform full validation or no validation. Valid values are FND_API.G_VALID_LEVEL_FULL (default)—and FND_API.G_VALID_LEVEL_NONE.

p_lpn

WMS_LICENSE_PLATE_NUMBERS stores the information for the fields of the LPN record that the user wants to modify.

p_source_type_id

Type identifier of the source transaction.

p_source_header_id

Header identifier of the source transaction.

p_source_name

Name of the source transaction.

p_source_line_id

Line identifier of the source transaction.

p_source_line_detail_id

Line detail identifier for the source transaction.

x_return_status

If the Create_LPN API succeeds, then the value is fnd_api.g_ret_sts_success. If an expected error occurs, then the value is fnd_api.g_ret_sts_error. If an unexpected error occurs, then the value is fnd_api.g_ret_sts_unexp_error.

x_msg_count

If one or more errors occur, then this value represents the number of error messages in the buffer.

x_msg_data

If only one error occurs, then this column contains the error message.

Oracle Warehouse Management Installation API

This API verifies that Oracle Warehouse Management is installed.

WMS_INSTALL. check_install

Parameter	Usage	Type	Required	Derived	Optional
p_organization_id	IN	Number			
x_return_status	OUT	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
x_msg_count	OUT	Number			
x_msg_data	OUT	Varchar2			

p_organization_id

Specific organization that the API verifies that Oracle Warehouse Management is enabled. If the field value is null, then the API verifies Oracle Warehouse Management at the site level only and not for any specific organization. This is more relaxed than passing a specific organization.

If Oracle Warehouse Management is installed, then the API returns TRUE. If Oracle Warehouse Management is not installed, then the API returns FALSE. Use the return value to determine whether Oracle Warehouse Management is installed. Do not use x_return_status for this purpose because the x_return_status value could be successful even if Oracle Warehouse Management is not installed. x_return_status is set to error when an error (such as a SQL error) occurs.

x_return_status

Return status that indicates success (S), error (E), unexpected error (U).

x_msg_count

Number of messages in a message list.

x_msg_data

If the number of messages in message list is 1, then this column contains the message list.

Radio Frequency Identifier APIs

Oracle Warehouse Management provides the flexibility of performing inbound-related and outbound-related transactions based on read events that are generated from a radio frequency identifier (RFID) reader. You can implement an RFID solution for an automatic receipt of an internal order or for a purchase order for which the supplier sends an ASN. Similarly, you can implement an RFID solution to automate the Truck Load or Truck Load and Ship outbound process.

Make sure that the RFID tag carries the LPN number. Using the LPN, Oracle Warehouse Management detects the internal order or purchase order for which a

receipt needs to be confirmed. For this reason a receipt can be confirmed only for purchase orders that have an ASN. Similarly, using the LPN, the system detects the sales order(s) that must be confirmed for shipment.

Process RFID Transaction API

You must register the API that is provided in this package with a middleware solution that communicates with the RFID reader, such as the Oracle Sensor Edge Server of Oracle Application Server. This API must invoked in response to an RFID-read event.

WMS_RFID_DEVICE_PUB.process_rfid_txn

Parameter	Usage	Type	Required	Derived	Optional
p_tagid	IN	CLOB	Y	N	N
P_tagdata	IN	CLOB	N	N	Y
P_portalId	IN	VARCHAR	Y	N	N
P_event_date	IN	DATE	Y	N	N
P_system_id	IN	VARCHAR	N	N	Y
P_statuschange	IN	NUMBER	N	N	Y
P_status	IN	NUMBER	N	N	Y
P_x	IN	NUMBER	N	N	Y
P_y	IN	NUMBER	N	N	Y
X_return_value	OUT	VARCHAR2			
X_return_msg	OUT	VARCHAR2			

P_tagid

Value on the RFID tag. This should correspond to a valid LPN in Oracle Warehouse Management. This parameter is currently not used by Oracle Warehouse Management.

P_tagdata

Unique identification of the RFID tag. Currently not used by Oracle Warehouse Management.

P_portalId

Identifier of the location from where this event was generated. This corresponds to a device setup in the Define Devices form in Oracle Warehouse Management.

P_event_date

Time stamp of the event. They are not currently used.

X_return_value

Indicates the outcome of processing the transaction. This parameter can have the following values: Success, Error, Warning.

X_return_mesg

Indicates any additional messages that are related to the outcome of processing the transaction.

WMS Read RFID Event API

You must register this API with Oracle Application Server and Oracle Sensor Edge Server if you need to drive a device as soon as Oracle Warehouse Management detects an RFID event. A sample usage of this API is to drive a warning light as soon as an RFID-tagged pallet is pushed through a portal with a mounted RFID reader.

WMS_RFID_DEVICE_PUB.WMS_READ_EVENT

Parameter	Usage	Type	Required	Derived	Optional
p_tagid	IN	CLOB	Y	N	N
P_tagdata	IN	CLOB	N	N	Y
P_portalId	IN	VARCHAR	Y	N	N
P_event_date	IN	DATE	Y	N	N
P_system_id	IN	VARCHAR	N	N	Y

Parameter	Usage	Type	Required	Derived	Optional
P_statuschange	IN	NUMBER	N	N	Y
P_status	IN	NUMBER	N	N	Y
P_x	IN	NUMBER	N	N	Y
P_y	IN	NUMBER	N	N	Y
X_return_value	OUT	VARCHAR2			
X_return_msg	OUT	VARCHAR2			

P_tagid

Value on the RFID tag. This value corresponds to a valid LPN in Oracle Warehouse Management.

P_tagdata

Unique identifier of the RFID tag. Currently not used by Oracle Warehouse Management.

P_portalId

Identifier of the location from where this event was generated. This value corresponds to the device setup in the Define Devices Form in Oracle Warehouse Management.

P_event_date

Time stamp of any of the following events: P_system_id, P_statuschange, p_status, p_statuschange, p_x, and p_y. These parameters are for future enhancements. These are not currently used.

X_return_value

Indicates the outcome that results from processing the transaction. This parameter can have the values Success, Error, Warning.

X_return_mesg

Indicates any additional messages that are related to the outcome of processing the transaction.

Get New Load Verification Threshold Value API

The Get New Load Verification Threshold Value API (WMS_RFID_EXT_PUB.GET_NEW_LOAD_VERIF_THRESHOLD) is a public API that provides extension code to the RFID functionality.

This API can be used as an extension API to obtain a load verification threshold value different from the one provided at the organization level. The purpose of this API is to override the simple percentage check functionality provided by the application.

This section explains how to use the Get New Load Verification Threshold Value API and how the API functions in Oracle Warehouse Management products.

API Features

The Get New Load Verification Threshold Value API enables you to obtain a new load verification threshold value based on own-logic, different from the one provided at the organization level. Table: WMS_LICENSE_PLATE_NUMBERS, WMS_LPN_CONTENTS.

When the internal API verifies the load, it makes a call to this custom API. If the return value from this custom API is non-void, then it makes use of this new load verification threshold value.

See also: Oracle Warehouse Management in the Electronic Technical Reference Manual (eTRM)

Functional Overview

This is a stub API. Each of these procedures has no implementation within them. You must provide your own custom implementation or skip them.

You must define what percentage of all tags on a pallet must be present in order for the pallet to be presumed fully present. This criterion is necessary because all the tags on the pallet might not be readable, based on the type of items contained in the pallet. Generally this criterion is provided as an organization parameter.

If you want to override the default criteria, which is just a percentage check, you can do so by using this custom API.

The input to this stub API is the list of all LPN identifiers seen by the verification process (that is, Oracle Warehouse Management translates between the EPC and the LPN) and the list of all unknown identifiers seen by the verification process (i.e., those tags that cannot be translated into either EPC or LPN, or that indicate LPNs not already known by Oracle Warehouse Management), unsegmented by inner or outer LPN.

The verification stub is then responsible for identifying the outer pallet, determining if enough cases are present and no unexpected tags seen, and providing a success or error back to the application.

Setting Up the API

The Get New Load Verification Threshold Value API is a stored PL/SQL procedure. Before using the API, set up and activate the following parameters. These parameters are passed by calling the internal API. Custom implementation must use these parameters.

- Organization ID
- Pallet LPN ID

Parameter Descriptions

The following table lists all parameters used by the public API WMS_RFID_EXT_PUB.GET_NEW_LOAD_VERIF_THRESHOLD. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

WMS_RFID_EXT_PUB.GET_NEW_LOAD_VERIF_THRESHOLD

Parameter Name	Usage	Type	Required?	Derived?
p_org_id	IN	NUMBER	Yes	No
P_pallet_lpn_id	IN	NUMBER	Yes	No
x_new_load_verif_threshold	OUT	NUMBER	Yes	No
x_return_status	OUT	Number	Yes	No

Note: This API is invoked by the WMS_RFID Internal API. The custom implementation must validate these values.

p_org_id

Organization ID of the transaction source.

p_pallet_lpn_id

LPN ID of the pallet used for the transaction. Using this value, the custom implementation can query the WMS_LICENSE_PLATE_NUMBERS table to obtain all relevant LPN list parameters for this transaction.

x_new_load_verif_threshold

New load verification threshold value which will be used instead of the default value specified for the organization. This must be returned as a percentage value.

x_return_status

Return status of the custom EPC generation. It can be one of the following values: Success, Error, Unexpected Error

Success: FND_API.G_RET_STS_SUCCESS

Error: FND_API.G_RET_STS_ERROR

Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of the API

Standard Validation

Because the values passed as parameters to these custom APIs are from the standard internal APIs, the validation occurs automatically.

Generating Custom EPC

Following gives the details of the validation for each of the parameters passed to this procedure.

p_org_id

Must be a valid organization that is defined in ORG_ORGANIZATION_DEFINITIONS.

p_pallet_lpn_id

Must be a valid LPN ID that is defined in the WMS_LICENSE_PLATE_NUMBERS table. Using the organization ID, LPN ID, and the WMS_LPN_CONTENTS table, all required details for the LPN for the current transaction can be obtained.

Other Validation

None.

Error Handling

If any validation fails, then the API returns the error status to the calling module. The WMS_RFID_EXT_PUB API processes the rows and reports the following values for every record:

Condition	PROCESS_STATUS	ERROR_MESSAGE
Success	5	null
Failure	4	<actual error message>

Close Truck and Ship Confirm API

This API performs the ship-confirmation after pallets are loaded to the truck without going through a user interface screen. It is used in conjunction with automatic loading of RFID-tagged pallets. You can use this API to communicate between Oracle Warehouse Management and a button that the end-user presses after pallets are loaded to the truck. This API verifies that all the LPNs in the specified location that are associated with the delivery or trip are in Loaded state. After this verification, the ship-confirmation transaction request is submitted.

WMS_SHIPPING_TRANSACTION_PUB.close_truck

Parameter	Usage	Type	Required	Derived	Optional
p_dock_door_id	IN	NUMBER	Y	N	N
P_organization_id	IN	NUMBER	Y	N	N
P_shipping_mode	IN	VARCHAR2	N	N	Y
P_commit_flag	IN	VARCHAR2	N	Y	Y
X_return_status	OUT	VARCHAR2			
X_return_msg	OUT	VARCHAR2			

P_dock_door_id

Identifier of the dock door where ship-confirmation will take place. This value corresponds to the valid dock-door ID in Oracle Warehouse Management.

P_organization_id

Identifier of a valid organization setup in Oracle Warehouse Management.

P_shipping_mode

Indicates the modes of shipment that must be processed. This column can contain the values NORMAL or DIRECT. NORMAL indicates the LPNs that have been staged as part of a normal pick-confirmation operation. DIRECT indicates the LPNs that are to be shipped using the Direct Ship functionality. If this parameter is not specified, then all available loaded LPNs at the dock door are shipped.

P_commit_mode

Indicates whether the transaction should be committed to the system.

X_return_status

Indicates the outcome of calling this API. It can contain any of the following values: Success (S), error (E), and warning (W).

X_return_msg

Message that this API can optionally return based on the value of the x_return_status parameter.

Electronic Product Code API

The EPC API (WMS_EPC_PUB) is a public API that enables you to generate non-standard custom or unimplemented EPC, implement your own logic to pick the company prefix and company prefix-index.

You can use this API as an extension API to support EPC generation of custom/unimplemented EPC types.

This section explains how to use the WMS_EPC_PUB API and how the API functions in Oracle Warehouse Management products.

EPC API Features

The WMS_EPC_PUB API consists of the following entities:

- Get Custom EPC: You can generate non-standard custom EPC or unimplemented EPC that can be stored in Oracle Warehouse Management and used for subsequent transactions.

  Table: WMS_EPC

- Get Custom Company Prefix value: Allows you to use your own logic for picking the company prefix value, if you want to use a different value than the one set at the organization level.

- Get Custom Company Prefix-Index value: Allows you to use your own logic for picking the company prefix-index value, if you want to use a different value than the one set at the organization level.

When the internal APIs generate the EPC for a transaction, they call each of these custom implementations. As such, the custom implementations have no code within them and return "success." If you want to use these APIs, then you must provide the custom implementation within each of these procedures and return the appropriate values. These values will be checked by internal APIs during EPC generation and used instead of the default values.

See also: Oracle Warehouse Management in the Electronic Technical Reference Manual (eTRM).

Functional Overview

- Get Custom EPC: Generate non-standard custom or unimplemented EPC.
- Get Custom Company Prefix value: Use your own logic for picking the company prefix value.
- Get Custom Company Prefix-index value: Use your own logic for picking the company prefix-index value

Each of these procedures has no implementation within them. You must provide your own custom implementation or skip these procedures. The EPC generation logic in the internal APIs are as follows:

First, the internal API calls the custom APIs for retrieving the Custom Company prefix and prefix-index values. If the custom APIs return a value, then they are used instead of the default values (specified at the Organization Parameters), that would be used otherwise.

Secondly, the internal API calls the custom API for retrieving the non-standard custom or implemented EPC. If you want to use non-standard EPC generation rules (EPC_GIAI_96, EPC_GRAI_96, EPC_SGLN_96, EPC_GID_96, EPC_GIAI_64, EPC_GRAI_64, EPC_SGLN_64, EPC_GID_64), then you must provide a custom implementation for generating the EPC. Note that the EPC generation rule must be defined appropriately in the application before using them.

No custom implementation is required in order to use the standard EPC generation rules (EPC_SGTIN_96, EPC_SSCC_96, EPC_SGTIN_64, EPC_SSCC_64, EPC_DOD_96, EPC_DOD_64). These rules are already available with the internal APIs.

If the custom API returns a valid EPC, then this value is used and corresponding updates to the WMS_EPC table are made.

Setting Up EPC API

Before using the EPC API, you must perform the following setup procedures:

- Define the EPC Generation Rule using the Oracle Database EPC generation utility.
- Define the label format with the required attributes. This label format must use the required EPC Generation Rule and appropriate filter value.

The EPC API is a stored PL/SQL procedure. Before using the API, set up and activate the following parameters:

- Organization ID
- EPC Category ID
- EPC Generation Rule type ID
- Label Request ID
- Filter Value

Parameter Descriptions

API WMS_EPC_PUB.GET_CUSTOM_EPC

The following table lists all parameters used by the public API WMS_EPC_PUB.GET_CUSTOM_EPC. All of the inbound and outbound parameters are listed. Additional information about these parameters follows the table.

WMS_EPC_PUB.GET_CUSTOM_EPC

Parameter Name	Usage	Type	Required?	Derived?
p_org_id	IN	VARCHAR2	Yes	No
p_category_id	IN	VARCHAR2	Yes	No
p_epc_rule_type	IN	VARCHAR2	Yes	No
p_filter_value	IN	NUMBER	Yes	No
p_label_request_id	IN	NUMBER	Yes	No

x_return_status	OUT	VARCHAR2	Yes	No
x_return_msg	OUT	VARCHAR2	Yes	No
x_epc	OUT	VARCHAR2	Yes	No

Note: This API is invoked by the WMS_EPC Internal API. The custom implementation must validate these values and generate the EPC accordingly.

p_org_id

Organization ID of the transaction source.

p_category_id

EPC Category ID as defined in the Oracle EPC Generation Utility.

p_epc_rule_type

EPC Generation rule type ID as defined in the Oracle EPC Generation Utility.

p_filter_value

Filter value to identify whether the label format is of the object type Pallet, Case, Inner-Pack, etc.

p_label_request_id

Provides all information for the current transaction from the WMS_LABEL_REQUESTS table.

x_return_status

Return status of the custom EPC generation. It can be one of the following values: Success (S), Error (E), Unexpected Error occurred (U)

Success: FND_API.G_RET_STS_SUCCESS

Error: FND_API.G_RET_STS_ERROR

Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_return_msg

Return message in case the custom EPC generation fails. By default, it returns NULL.

x_epc

Generated EPC returned back to the caller as a HEX value. Without custom implementation, this returns NULL by default.

WMS_EPC_PUB.GET_CUSTOM_COMPANY_PREFIX

The following table lists all parameters used by the public API WMS_EPC_PUB.GET_CUSTOM_COMPANY_PREFIX. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

WMS_EPC_PUB.GET_CUSTOM_COMPANY_PREFIX

Parameter Name	Usage	Type	Required?	Derived?
p_org_id	IN	VARCHAR2	Yes	No
p_label_request_id	IN	NUMBER	Yes	No
x_company_prefix	OUT	VARCHAR2	Yes	No
x_return_status	OUT	VARCHAR2	Yes	No

Note: This API is invoked by the WMS_EPC Internal API. The custom implementation must validate these values and generate the EPC accordingly.

p_org_id

Organization ID of the transaction source.

p_label_request_id

Provides all information for the current transaction from WMS_LABEL_REQUESTS table.

x_company_prefix

Custom generated company prefix which would be used instead of the default value.

x_return_status

Return status of the custom EPC generation. It can be one of the following values: Success (S), Error (E), Unexpected Error occurred (U)

Success: FND_API.G_RET_STS_SUCCESS

Error: FND_API.G_RET_STS_ERROR

Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

WMS_EPC_PUB.GET_CUSTOM_COMP_PREFIX_INDEX

The following table lists all parameters used by the public API WMS_EPC_PUB.GET_CUSTOM_COMP_PREFIX_INDEX. All of the inbound and outbound parameters are listed. Additional information on these parameters follows the table.

WMS_EPC_PUB.GET_CUSTOM_COMP_PREFIX_INDEX

Parameter Name	Usage	Type	Required?	Derived?
p_org_id	IN	VARCHAR2	Yes	No
p_label_request_id	IN	NUMBER	Yes	No
x_comp_prefix_index	OUT	VARCHAR2	Yes	No
x_return_status	OUT	VARCHAR2	Yes	No

Note: This API is invoked by the WMS_EPC Internal API. The custom implementation must validate these values and generate the EPC accordingly

p_org_id

Organization ID of the transaction source.

p_label_request_id

Provides all information for the current transaction from the WMS_LABEL_REQUESTS table.

x_comp_prefix_index

Custom-generated company prefix that is used instead of the default value.

x_return_status

Return status of the custom EPC generation. It can be one of the following values: Success (S), Error (E), Unexpected Error occurred (U)

Success: FND_API.G_RET_STS_SUCCESS

Error: FND_API.G_RET_STS_ERROR

Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

Validation of WMS_EPC_PUB

Standard Validation

Because the values passed as parameters to these custom APIs are from the standard internal APIs, the validation occurs automatically.

Generating Custom EPC

Following gives the details of the validation for each of the parameters passed to this procedure.

p_org_id

Must be a valid organization defined in ORG_ORGANIZATION_DEFINITIONS

p_category_id

Must be valid EPC Category ID defined in the Oracle Database EPC generation utility. This is generated when the label format is created and is fetched from WMS_LABEL_FORMATS table by the caller.

p_epc_rule_type_id

Must be valid EPC rule type ID defined in the Oracle Database EPC generation utility. This is generated when the EPC Generation Rule is defined and is fetched from WMS_LABEL_FORMATS table by the caller.

p_label_request_id

Must be a valid label_request_id in the WMS_LABEL_REQUESTS table. This can be used to retrieve all necessary information regarding the transaction.

Generating Custom Company Prefix

Same validations as discussed for Custom EPC Generation.

Generating Custom Company Prefix-index

Same validations as discussed for Custom EPC Generation.

Other Validation

None.

Error Handling

If any validation fails, then the API returns the error status to the calling module. The WMS_EPC_PUB API processes the rows and reports the following values for every record:

Condition	PROCESS_STATUS	ERROR_MESSAGE
Success	5	null
Failure	4	<actual error message>

Crossdock Customization API

This section explains how to use the WMS_XDOCK_CUSTOM_APIS_PUB API and how the API functions in Oracle Warehouse Management.

The Crossdock Customization API (WMS_XDOCK_CUSTOM_APIS_PUB) consists of several stubbed-out PL/SQL procedures that you can use to customize the way crossdocking is done in Oracle Warehouse Management. These procedures are invoked from the crossdock pegging engine (WMS_XDOCK_PEGGING_PUB), and if implemented, override the default behavior of the pegging engine. Crossdock pegging refers to the process of matching supply and demand lines for both planned and opportunistic crossdocking.

Crossdock Customization API Features

The Crossdock Customization API consists of the following entities:

- Crossdock criteria: These determine which sources of supply or demand are valid, as well as the time intervals to be taken into consideration before creating a crossdock peg (reservation). Crossdock criteria can be specified on the pick release form (planned crossdock), on the Organization Parameters form (opportunistic crossdock), or can be assigned using the Rules Workbench (both planned and opportunistic crossdocking). A custom API exists in WMS_XDOCK_CUSTOM_APIS_PUB that lets you assign crossdock criteria for planned crossdocking based on custom logic.

  Tables: WMS_CROSSDOCK_CRITERIA_B (base table), WMS_CROSSDOCK_CRITERIA_TL (translated columns)

- Supply lines: These represent valid sources of supply for planned crossdocking. Based on the crossdock criterion being used, the crossdock pegging engine will retrieve details about available supply sources (such as approved purchase orders and ASNs) from multiple database tables. This data is then made available as a PL/SQL table for customized sorting methods.

- Demand lines: These represent valid sources of demand for opportunistic crossdocking such as sales orders, internal orders, or backordered component demand. This data can also be sorted using custom code.

See also: *Oracle Warehouse Management User's Guide* and *Oracle Warehouse Management Implementation Guide*.

Functional Overview

The Crossdock Customization API provides the following stubbed out PL/SQL procedures that you can implement as needed:

- Get_Crossdock_Criteria: This API is used to define a custom method of selecting the crossdock criterion to use for planned crossdocking. If it is implemented, the planned crossdocking pegging process uses this API instead of the Oracle Warehouse Management rules engine to determine what criterion to use for each demand line being planned.

- Get_Expected_Time: Crossdock pegging takes into consideration the expected receipt time and ship times of supply and demand lines, respectively. You can use this API to customize the way that receipt and ship time are calculated.

- Get_Expected_Delivery_Time: Opportunistic crossdocking looks for existing deliveries to which the current demand line can be merged, if they are not already assigned to a delivery. This API can be used to customize the way the expected ship time is calculated for a delivery.

- Sort_Supply_Lines: Implement this API to customize the order in which supply lines are sorted for planned crossdocking. In addition, set the crossdocking goal to Custom to ensure that the crossdock pegging engine calls this API.

- Sort_Demand_Lines: Implement this API to customize the order in which demand lines are sorted for opportunistic crossdocking. In addition, set the crossdocking goal to Custom to ensure that the crossdock pegging engine calls this API.

Setting Up the Crossdock Customization API

In order to implement and use the custom APIs in this package, you must perform the necessary setups for crossdocking within Oracle Warehouse Management. This includes defining crossdock criteria, enabling crossdock through the Organization Parameters form (for opportunistic crossdock only), defining crossdock operation plans (optional), and so on. Refer to the *Oracle Warehouse Management User's Guide* and the *Oracle Warehouse Management Implementation Guide* for more information.

After crossdocking is set up and the APIs are implemented, executing any crossdock flow through the application results in the custom APIs being executed by the crossdock pegging engine.

Parameter Descriptions

The following table lists all parameters used by the public API Crossdock Customization. All of the inbound and outbound parameters for each procedure are listed. Additional information about these parameters follows the table.

WMS_XDOCK_CUSTOM_APIS_PUB.GET_CROSSDOCK_CRITERIA

Parameter Name	Usage	Type	Required?	Derived?

p_wdd_release_record	IN	PL/SQL record	Yes	No
x_return_status	OUT	Varchar2	Yes	No
x_msg_count	OUT	Number	No	No
x_msg_data	OUT	Varchar2	No	No
x_api_is_implemented	OUT	Boolean	Yes	No
x_crossdock_criteria_id	OUT	Number	No	Yes

p_wdd_release_record

Delivery detail line for which you are trying to determine a valid planned crossdock criterion. The demand line record contains relevant information such as org, item, customer ID, project, and task. This record is the current record being processed by the calling API (WMS_XDOCK_PEGGING_PUB.PLANNED_CROSS_DOCK).

The parameter p_wdd_release_record is of type WSH_PR_CRITERIA.relRecTyp. It is described in the package specification WSH_PR_CRITERIA (WSHPRCRS.pls) owned by Oracle Shipping Execution. The record type contains a subset of the columns in the database table WSH_DELIVERY_DETAILS.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages that the API encounters.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_is_api_implemented

Indicates to the calling API (WMS_XDOCK_PEGGING_PUB.PLANNED_CROSS_DOCK) whether a custom GET_CROSSDOCK_CRITERIA is implemented. If it is implemented, the planned crossdock pegging process does not call the Oracle Warehouse Management rules engine to find the crossdock criterion to use for the current demand line.

x_crossdock_criteria_id

The output crossdock criteria ID of type Planned. If no valid crossdock criteria could be determined, then a NULL value is returned. If a value of -1 is returned, this will be interpreted as Not Implemented. The pegging logic then call the rules engine to determine valid crossdock criteria instead.

WMS_XDOCK_CUSTOM_APIS_PUB.GET_EXPECTED_TIME

Parameter Name	Usage	Type	Required?	Derived?
p_source_type_id	IN	Number	Yes	No
p_source_header_id	IN	Number	Yes	No
p_source_line_id	IN	Number	Yes	No
p_source_line_detail_id	IN	Number	Yes	No
p_supply_or_demand	IN	Number	Yes	No
p_crossdock_criterion_id	IN	Number	Yes	No
p_dock_schedule_method	IN	Number	Yes	No
x_return_status	OUT	Varchar2	Yes	No
x_msg_count	OUT	Number	No	No
x_msg_data	OUT	Varchar2	No	No
x_api_is_implemented	OUT	Boolean	Yes	No
x_dock_start_time	OUT	Date	No	Yes
x_dock_mean_time	OUT	Date	No	Yes

x_dock_end_time	OUT	Date	No	Yes
x_expected_time	OUT	Date	No	Yes

p_source_type_id

Source type of supply or demand line. This corresponds to the lookup Reservation_Types used by reservations. Example: 1 = PO, 2 = Sales Order, 5 = WIP, 7 = Internal Req, 8 = Internal Order

p_source_header_id

Source Header ID, such as PO Header ID, RCV Shipment Header ID, and OE Order Header ID. This corresponds to what reservations inserts for various supply or demand types.

p_source_line_id

Source Line ID, such as PO Line Location ID, RCV Shipment Line ID, and OE Order Line ID. This corresponds to what reservations inserts for various supply or demand types.

p_source_line_detail_id

Source Line Detail ID, such as Delivery Detail ID. This corresponds to what reservations inserts for various supply or demand types.

p_supply_or_demand

Line type (supply or demand) you want to get the expected receipt or ship time for: 1 = Supply 2 = Demand.

p_crossdock_criterion_id

Crossdock criterion ID, if available. This determines how to calculate the expected receipt or ship time in case a dock appointment exists. Dock appointments are an interval with a start and end time so it is possible to use the begin, mean, or end time.

p_dock_schedule_method

Dock schedule method for the crossdock criterion (if value is passed). This will correspond to either the supply_schedule_method or demand_schedule_method depending on the line type (supply or demand).

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages the API encounters.

x_msg_data

Displays the error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_is_api_implemented

Indicates to the calling API (WMS_XDOCK_PEGGING_PUB.GET_EXPECTED_TIME) whether a custom GET_EXPECTED_TIME API has been implemented. If it is implemented, then the crossdock pegging process skips over the standard process used to determine expected receipt or shipment time.

x_dock_start_time

The dock appointment start time (if it exists).

x_dock_mean_time

The dock appointment mean time (if it exists).

x_dock_end_time

The dock appointment end time (if it exists).

x_expected_time

The expected receipt or ship time for the line. If the crossdock criterion is passed and a dock appointment exists, then expected time is based on the dock start and end time using the supply_schedule_method or demand_schedule_method. If a dock appointment exists but a crossdock criterion is not passed, then the value is NULL.

WMS_XDOCK_CUSTOM_APIS_PUB.GET_EXPECTED_DELIVERY_TIME

Parameter Name	Usage	Type	Required?	Derived?
p_delivery_id	IN	Number	Yes	No
p_crossdock_criterion_id	IN	Number	Yes	No

p_dock_schedule_method	IN	Number	Yes	No
x_return_status	OUT	Varchar2	Yes	No
x_msg_count	OUT	Number	No	No
x_msg_data	OUT	Varchar2	No	No
x_api_is_implemented	OUT	Boolean	Yes	No
x_dock_appointment_id	OUT	Number	No	Yes
x_dock_start_time	OUT	Date	No	Yes
x_dock_end_time	OUT	Date	No	Yes
x_expected_time	OUT	Date	No	Yes

p_wdd_release_record

Delivery detail line record that you are trying to crossdock. The parameter p_wdd_release_record is of type WSH_PR_CRITERIA.relRecTyp and is described in the package specification WSH_PR_CRITERIA (WSHPRCRS.pls) owned by Oracle Shipping Execution. The record type contains a subset of the columns in the database table WSH_DELIVERY_DETAILS.

p_prioritize_documents

Flag that indicates if the supply source documents should be prioritized. The possible values are: 1 = Yes, 2 = No. If the documents must be prioritized, then the supply lines in the input p_shopping_basket_tb will already be sorted by the supply source types in the order that they should be consumed. In this case, the custom API should sort the supply lines by the source types to maintain the relative ordering of the supply documents.

p_shopping_basket_tb

Table of valid supply lines for crossdocking. The supply lines all lie within the crossdock window of the demand and are valid source types for crossdocking. The supply line records stored in this table are the row records from the global temp table, wms_xdock_pegging_gtmp.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates that number of error messages that the API encounters.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_is_api_implemented

Indicates to the calling API (WMS_XDOCK_PEGGING_PUB.PLANNED_CROSS_DOCK) whether a custom sorting API has been implemented.

x_sorted_order_tb

Output table that indicates the order in which the supply lines in p_shopping_basket_tb must be sorted. This table should be indexed by consecutive integers starting with 1 (for example, 1, 2, 3, 4, ...) The table entry with index 1 will contain the pointer to the index value in p_shopping_basket_tb corresponding to the first supply line record that should be consumed. The same logic applies to index 2, 3, 4, ... in x_sorted_order_tb. If for some reason the custom logic does not want to use a supply line that exists in p_shopping_basket_tb, then do not include that entry in x_sorted_order_tb. This output table can have less than or an equal amount of entries as p_shopping_basket_tb. It can never have more. Additionally, multiple entries in x_sorted_order_tb must not point to the same index value. If an invalid sorted order table is returned, then the custom logic is not used to sort the shopping basket table.

WMS_XDOCK_CUSTOM_APIS_PUB.SORT_DEMAND_LINES

Parameter Name	Usage	Type	Required?	Derived?
p_move_order_line_id	IN	Number	Yes	No
p_prioritize_documents	IN	Number	Yes	No
p_shopping_basket_tb	IN	PL/SQL Table	Yes	No

x_return_status	OUT	Varchar2	Yes	No
x_msg_count	OUT	Number	No	No
x_msg_data	OUT	Varchar2	No	No
x_api_is_implemented	OUT	Boolean	Yes	No
x_sorted_order_tb	OUT	PL/SQL Table	Yes	Yes

p_move_order_line_id

Move order line ID for the supply you want to crossdock.

p_prioritize_documents

Flag that indicates if the demand source documents must be prioritized. The possible values are: 1 = Yes, 2 = No. If the documents are to be prioritized, the demand lines in the input p_shopping_basket_tb will already be sorted by the demand source types in the order they should be consumed. In this case, the custom API should sort the demand lines by the source types to maintain the relative ordering of the demand documents.

p_shopping_basket_tb

Table of valid demand lines for crossdocking. The demand lines all lie within the crossdock window of the demand and are valid source types for crossdocking. The demand line records stored in this table are from the global temp table, wms_xdock_pegging_gtmp.

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages that the API encounters.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the actual message.

x_is_api_implemented

Indicates to the calling API (WMS_XDOCK_PEGGING_PUB.OPPORTUNISTIC_CROSS_DOCK) whether a custom sorting API has been implemented.

x_sorted_order_tb

Output table that indicates the order in which the demand lines in p_shopping_basket_tb must be sorted. This table should be indexed by consecutive integers starting with 1 (for example, 1, 2, 3, 4, ...). The table entry with index 1 will contain the pointer to the index value in p_shopping_basket_tb corresponding to the first demand line record that should be consumed. The same logic applies to index 2, 3, 4, ... in x_sorted_order_tb. If for some reason the custom logic does not want to use a demand line that exists in p_shopping_basket_tb, then do not include that entry in x_sorted_order_tb. This output table can have less than or an equal amount of entries as p_shopping_basket_tb. It can never have more. Additionally, multiple entries in x_sorted_order_tb must not point to the same index value. If an invalid sorted order table is returned, then the custom logic is not used to sort the shopping basket table.

Validation of Crossdock Customization API

Standard Validation

Because this API consists of stubs that you need to implement, no validations are included.

Other Validation

None.

Error Handling

Some basic exception handling is included in the stubs provided, but you must implement additional error handling, as needed.

Rules Engine Custom API

This section explains how to use the Rules Engine Custom API (WMS_RE_CUSTOM_PUB API) and how the API functions in Oracle Warehouse Management System.

This API is a collection of customizable stub procedures and functions that are called

when the Oracle Warehouse Management rules engine runs.

Rules Engine Custom API Features

The Rules Engine Custom API consists of the following entities:

- GetTotalLocationCapacity: Calculates and returns the total capacity of a location (subinventory or subinventory and locator) according to your specifications.
- GetOccupiedLocationCapacity: Calculates and returns the occupied capacity of a location (subinventory or subinventory and locator) according to your specifications.
- GetAvailableLocationCapacity: Calculates and returns the available capacity of a location (subinventory or subinventory and locator) according to your specifications.
- SearchForStrategy: Searches for an Oracle Warehouse Management strategy, rule, or value assignments to a custom business object according to your specifications.

Functional Overview

Using the following stub procedures and functions, you can write and implement custom capacity calculations, custom strategy, and rule search logic. The first three functions compute locator capacity during putaway based on the your logic. You can use the fourth procedure to implement your custom strategy or rule search for picking, putaway, and cost group rules.

- GetTotalLocationCapacity: Calculates and returns the total capacity of a location based on the subinventory and locator that you supply.
- GetOccupiedLocationCapacity: Calculates and returns the occupied capacity of a location based on the input that you supply. This function is available to use within putaway.
- GetAvailableLocationCapacity: Calculates and returns the available capacity of a location based on the supplied input according to your specifications. This function is available to use within putaway.
- SearchForStrategy: Searches for an Oracle Warehouse Management strategy, rule, or value assignment to a custom business object according to your specifications.

Setting Up the Rules Engine Custom API

This API is a placeholder for custom logic to implement the above functions. The only requirement is to write the code with the proper return values and error handling routines to avoid any run-time errors.

If you want to write your own capacity calculations, then select "custom algorithm" for the quantity function in the Rule Definition form for putaway rules.

There are no special requirements for the SearchForStrategy API.

Parameter Descriptions

The following table lists all parameters used by the public GetTotalLocationCapacity API. All of the inbound and outbound parameters are listed. Additional information about these parameters follows the table.

WMS_RE_CUSTOM_PUB.GetTotalLocationCapacity

Parameter Name	Usage	Type	Required?	Derived?
p_organization_id	In	Number	Yes	No
subinventory_code	In	VARCHAR2	Yes	No
p_locator_id	In	Number	Yes	No
p_inventory_item_id	In	Number	Yes	No
p_transaction_uom	In	VARCHAR2	Yes	No
return value	Out	Number	No	Yes

p_organization_id

The identification number for the organization that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

P_subinventory_code

The subinventory code that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

p_locator_id

The identification number for the locator that is referenced in the item locator record.

Default Value: NULL

p_inventory_item_id

The identification number for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_quantity

The quantity that is expressed in the transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_uom

The transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

return value

The return value that is the derived value of the locator capacity in the item locator record.

Default Value: 0

WMS_RE_CUSTOM_PUB.GetOccupiedLocationCapacity

Parameter Name	Usage	Type	Required?	Derived?
p_organization_id	In	Number	Yes	No
subinventory_code	In	VARCHAR2	Yes	No
p_locator_id	In	Number	Yes	No
p_inventory_item_id	In	Number	Yes	No
p_transaction_uom	In	VARCHAR2	Yes	No
return value	Out	Number	No	Yes

p_organization_id

The identification number for the organization that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

P_subinventory_code

The code for the subinventory that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

p_locator_id

The identification number for the locator that is referenced in the item locator record.

Default Value: NULL

p_inventory_item_id

The identification number for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_quantity

The quantity that is expressed in the transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_uom

The transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

return value

The return value that is the derived value of the locator capacity in the item locator record.

Default Value: 0

WMS_RE_CUSTOM_PUB.GetAvailableLocationCapacity

Parameter Name	Usage	Type	Required?	Derived?
p_organization_id	In	Number	Yes	No

P_subinventory_code	In	VARCHAR2	Yes	No
p_locator_id	In	Number	Yes	No
p_inventory_item_id	In	Number	Yes	No
p_transaction_quantity	In	Number	Yes	No
p_transaction_uom	In	VARCHAR2	Yes	No
return value	Out	Number	No	Yes

p_organization_id

The identification number for the organization that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

P_subinventory_code

The code for the subinventory that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

p_locator_id

The identification number for the locator that is referenced in the item locator record.

Default Value: NULL

p_inventory_item_id

The identification number for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_quantity

The quantity that is expressed in the transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_uom

The transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

return value

The return value that is the derived value of the locator capacity in the item locator record.

Default Value: 0

WMS_RE_CUSTOM_PUB.GetRemainingLocationCapacity

Parameter Name	Usage	Type	Required?	Derived?
p_organization_id	In	Number	Yes	No
P_subinventory_code	In	VARCHAR2	Yes	No
p_locator_id	In	Number	Yes	No
p_inventory_item_id	In	Number	Yes	No
p_transaction_quantity	In	Number	Yes	No
p_transaction_uom	In	VARCHAR2	Yes	No
return value	Out	Number	No	Yes

p_organization_id

The identification number for the organization that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

P_subinventory_code

The code for the subinventory that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

p_locator_id

The identification number for the locator that is referenced in the item locator record.

Default Value: NULL

p_inventory_item_id

The identification number for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_quantity

The quantity that is expressed in the transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_NUM

p_transaction_uom

The transactional unit of measure for the item that is referenced in the item locator record.

Default Value: FND_API.G_MISS_CHAR

return value

The return value that is the derived value of the locator capacity in the item locator record.

Default Value: 0

WMS_RE_CUSTOM_PUB.SearchForStrategy

Parameter Name	Usage	Type	Required?	Derived?
p_init_msg_list	In	VARCHAR2	No	Yes
x_return_status	Out	VARCHAR2	No	Yes
,x_msg_count	Out	Number	No	Yes
x_msg_data	Out	VARCHAR2	No	Yes
p_transaction_temp_id	In	Number	Yes	No

p_type_code	In	VARCHAR2	Yes	No
x_return_type	Out	VARCHAR2	Yes	Yes
x_return_type_id	Out	Number	Yes	Yes

p_init_msg_list

Requests that the API initialize the message list on your behalf. If the x_msg_count is greater than 1, then you must retrieve the list of messages using the call FND_MSG_PUB.GET. The values are:

- p_msg_index => I
- p_encoded => F
- p_data => 1_message
- p_msg_index_out => 1_msg_index_out

  where 1_message and 1_msg_index_out are local variables of types Varchar2 (2000 and Number, respectively)
- Default Value: FND_API.G_FALSE

p_return_values

Requests that the API send back the values on your behalf.

Default Value: FND_API.G_FALSE

x_return_status

Requests that the API return the status of the data for you after it completes its function. Valid values include:

- Success: FND_API.G_RET_STS_SUCCESS
- Error: FND_API.G_RET_STS_ERROR
- Unexpected Error: FND_API.G_RET_STS_UNEXP_ERROR

x_msg_count

Indicates the number of error messages that the API encountered.

x_msg_data

Displays error message text. If the x_msg_count is equal to 1, then this contains the

actual message.

p_transaction_temp_id

The identification number for the move order line that is used to determine all the values of an item in the move order line to compare to match the values in the strategy assignment record.

Default Value: FND_API.G_MISS_NUM

p_type_code

The type code is a rule type identifier that is passed to the API. Based on the value of the field , it determines whether the strategy or rule search is for picking, putaway, or cost group rules.

Possible values:

- 1 – Putaway rules
- 2 – Picking rules
- 3 – Cost group rules

x_return_type

Returns the information type of match that was found for the calling API - Rule / Strategy /Value.

Possible values:

- R – Rule
- S – Strategy
- V – Value

For picking and putaway rules, R and S apply. For cost group rules, all three values apply.

x_return_type_id

Returns the rule, strategy, or value identifier to the calling API, based on the x_retun_type_id.

If the value iş:

- rule_id, then the return type is R.
- strategy_id, then the return type is S.
- costgroup_id, then the return type is V.

Validation of Rules Engine Custom API

Oracle Warehouse Management does not validate any columns in the Rules Engine Custom API. Because it is a custom code, you must write your own validations and error handling code.

16

Receiving Open Interface

This chapter covers the following topics:

- Receiving Open Interface
- Functional Overview
- Receiving Transaction Processor Activities
- ASN Quantity Updates
- Cascading Transaction Quantities for ASNs and Receipts
- Setting Up the Receiving Open Interface
- Inserting into the Receiving Open Interface Table
- Receiving Headers Interface Table Description
- Receiving Transactions Interface Table Description
- Oracle Warehouse Management License Plate Number Interface Table Description
- Derived Data
- Optional Data
- Validation
- Debugging
- Resolving Failed Receiving Open Interface Rows
- Receiving Open Interface Details for Advanced Shipment Notice Import and Receipt Transactions
- Receiving Open Interface Data Details for Post Receipt Transactions
- Sample Scripts
- Stuck Transaction Scenarios and Scripts
- Oracle Warehouse Management LPN Interface Table Supplemental Information

Receiving Open Interface

You use the Receiving Open Interface to process and validate receipt data that comes from sources other than the Receipts window in Oracle Purchasing. These sources include:

- Receipt information from other Oracle applications or legacy systems
- Brocades and other receiving information from scanners and radio frequency devices
- Advance Shipment Notices (ASNs) from suppliers

The Receiving Open Interface maintains the integrity of the new data as well as the receipt data that resides in Oracle Purchasing.

The Receiving Open Interface does not support:

- Movement statistics
- Dynamic locators

The following table identifies the supported transactions:

Receiving Interface Supported Transactions

Transaction	ASN-PO	Internal Order	Interorg Transfer	RMA
RECEIVE	Yes	Yes	Yes	Yes
TRANSFER	Yes	Yes	Yes	Yes
ACCEPT/REJECT	Yes	Yes	Yes	Yes
DELIVER to Inventory	Yes	Yes	Yes	Yes
DELIVER to Expense	Yes	Yes	NA	NA
DELIVER to Shop Floor	Yes	Yes	NA	NA

Transaction	ASN-PO	Internal Order	Interorg Transfer	RMA
RETURN TO RECEIVING	Yes	No	No	NA
RETURN TO VENDOR	Yes	No	No	NA
RETURN TO CUSTOMER	NA	No	No	Yes
+ CORRECT to RECEIVE	Yes	No	Yes	No
- CORRECT to RECEIVE	Yes	No	Yes	Yes
CORRECT to TRANSFER	Yes	No	Yes	Yes
CORRECT to ACCEPT/REJECT	Yes	No	Yes	Yes
CORRECT to DELIVER to Inventory	Yes	No	No	No
CORRECT to DELIVER to Expense	Yes	No	NA	NA
CORRECT to DELIVER to Shop Floor	Yes	No	NA	NA
CORRECT to RETURN TO RECEIVING	No	No	NA	NA
CORRECT to RETURN TO VENDOR	Yes *	NA	NA	NA

Transaction	ASN-PO	Internal Order	Interorg Transfer	RMA
CORRECT to RETURN TO CUSTOMER	NA	NA	NA	Yes

Note: The Receiving Open Interface does not support corrections to returns for global procurement transactions, nor does it support corrections for drop shipments that use transaction flows for accounting.

The Receiving Open Interface does not support:

- Corrections to return to receiving
- Corrections to deliver, except when the source document is an ASN or a purchase order (PO)
- Positive corrections to receipts against a return material authorization (RMA)
- Returns or corrections against internal orders and interorganization transfers
- Unordered receipts and matching to unordered receipts
- Additions to receipt transactions
- Inspection transactions that interface with Oracle Quality

Functional Overview

Within the Receiving Open Interface, receipt data is validated for compatibility with Oracle Purchasing. There are two Receiving Open Interface tables: RCV_HEADERS_INTERFACE and RCV_TRANSACTIONS_INTERFACE.

Electronic Data Interchange Transaction Types

The Receiving Open Interface supports these Electronic Data Interchange (EDI) transaction types:

- Inbound advance shipment notices (ASN) (ANSI X12 856 or EDIFACT DESADV) original (new), canceled, and test ASNs
- Inbound ASNs with billing information (ANSI X12 857) original (new), canceled,

and test ASNs.

- Outbound Application Advices (ANSI X12 824 or EDIFACT APERAK)

An ASN is transmitted through EDI from a supplier to communicate to the receiving organization that a shipment is coming. For a detailed description of the ASN process, ASN types, Application Advices, and the effects of ASNs on purchasing supply, see Advance Shipment Notices (ASNs), *Oracle Purchasing User's Guide*.

Supported Transaction Features

The Receiving Open interface supports creation of ASN receipts, deliveries returns to vendor, returns to customers, corrections, and transfers. The Receiving Open Interface supports all of these transactions for all source documents including purchase orders, RMAs, interorg transfers and internal orders. The Receiving Open Interface does not support unordered receipts or any transactions for unordered receipts. The Receiving Open Interface supports these features:

- Receiving subinventory and locator: You can identify the receiving location, receiving subinventory, and receiving locator in the Receiving Open Interface. You can enter the subinventory, locator, or locator_id for receive, transfer, and return to vendor from receipt transactions, and the system transfers this information to other transactions. In the subsequent transactions, the receiving subinventory becomes the from subinventory, and the receiving locator becomes the from locator.

- Parent-child transactions: The system can create multiple transactions at the same time. For example, you can receive five units, deliver three units, and correct one unit from the delivery by inserting all the transactions at the same time in the rcv_transactions_interface. The parent_interface_transaction_id links these rows. The interface_transaction_id of the receive rcv_transactions_interface row is the parent_interface_transaction_id of the deliver row. The interface_transaction_id of the deliver row is the parent_interface_transaction_id of the correct row.

- Lot and serial: The Receiving Open Interface supports lot and serial-based delivery transactions and some receiving-based transactions, such as receive, transfer, and return to vendor from receiving. You can define lot and serial information for lot or serial-controlled items on all receiving transactions that you import through the Receiving Open Interface. These transactions include: receipts, transfers, inspections, deliveries, returns, and corrections. When you receive an item, lot and serial information is optional, but upon delivery, lot and serial information is required. However, if you specified lot and serial information on previous transaction then the system requires that you enter this information on all subsequent transactions. .

    - Lot support: Multiple rows can exist in the mtl_transaction_lots_interface for each row in the rcv_transactions_interface. One row must exist for each lot number in the mtl_transaction_lots_interface (mtli) where

mtli.product_transacton_id = rti.interface_transaction_id and mtli.product_code = RCV.

- Serial Support: Multiple rows can exist in the mtl_serial_numbers_interface for each row in the rcv_transactions_interface if the mtl_serial_numbers_interface.product_transaction_id = rti.interface_transaction_id or if the msni.transaction_interface_id = mtli.serial_transaction_temp_id if the item is lot and serial controlled.

  Important: The system does not support lot and serial functionality for receiving transactions through the desktop user interface. For example, if you use the Receiving Open Interface to perform a receiving transaction and then attempt to use the desktop user interface to perform a delivery transaction, the system returns an error message. If you enter lot and serial information in the Receiving Open Interface during receipt transaction, you must use the Receiving Open Interface, Oracle Mobile Supply Chain Applications, or Oracle Warehouse Management mobile interfaces to perform all subsequent transactions.

- LPN: You can use the Receiving Open Interface to perform the LPN actions pack, unpack, nest, and transfer on receiving transactions. LPN information is optional, but once you enter LPN information in a receiving transaction, the LPN remains with the material unless you change it. Suppliers can also include lot, serial, and LPN information in an ASN. The system automatically transfers this information to subsequent transactions.

  You can enter the LPN_id or the Transfer_LPN_id, depending upon the transaction. You can also provide a new LPN in the wms_lpn_interface table. You connect this row to the rcv_transactions_interface by populating the lpn_group_id in the rcv_transactions_interface as the source_group_id in the wms_lpn_interface table.

  Important: In an inventory organization, no LPNs can have the same name and an LPN can exist in only one location at any given time.

  The Receiving Open Interface supports LPNs in both Oracle Warehouse Management-enabled organizations and non-Oracle Warehouse Management-enabled organizations. You can use all LPN functionality in receiving until you deliver the LPN to inventory in a non-Oracle Warehouse Management-enabled organization. At delivery, in a non-Oracle Warehouse Management-enabled organization, the system automatically unpacks the LPNs and delivers the contents as loose material. The desktop receiving user interface does not support LPN use. If you import upstream receiving transactions through the Receiving Open Interface and use LPNs, then the system automatically unpacks

any subsequent transactions that you perform in the desktop user interface.

Validation Overview

The Receipt window derives, defaults, and validates the receipt data that you enter in the Receipts window. The receiving transaction processor derives, defaults, and validates most receipt data that you import through the Receiving Open Interface. The pre-processor is a component of Receiving Transaction Processor that validates and defaults the missing data. This is similar to the defaulting and validation that happens on the receiving desktop user interface when transaction is saved.

The following sections provide an overview of what the Receiving Open Interface does for each transaction:

Header-Level Validation

You use EDI or another program to load the receipt data into the RCV_HEADERS_INTERFACE and RCV_TRANSACTIONS_INTERFACE tables. The processor selects unprocessed rows in the RCV_HEADERS_INTERFACE table for preprocessing. It preprocesses rows with a PROCESSING_STATUS_CODE of PENDING and a VALIDATION_FLAG of Y. The processor derives or defaults any missing receipt header information in the RCV_HEADERS_INTERFACE table. For example, if you provide a TO_ORGANIZATION_CODE, the processor defaults the correct TO_ORGANIZATION_ID. The processor validates the receipt header information in the RCV_HEADERS_INTERFACE table to ensure the integrity of the information. The system imports only validated header information into the Oracle Purchasing tables. If the system does not detect fatal errors at the header level, then the Receiving Transaction Processor selects all the lines in the RCV_TRANSACTIONS_INTERFACE table associated with each header and calls the processor to perform line-level processing.

Line-Level Validation

- The processor derives and defaults any missing receipt line information in the RCV_TRANSACTIONS_INTERFACE table.
- The processor validates the receipt line information to ensure the integrity of the information.
- For successfully validated lines, the processor deletes the original RCV_TRANSACTIONS_INTERFACE line and creates the new, validated lines. The system sometimes creates two or more validated rows in the RCV_TRANSACTIONS_INTERFACE table to represent the original imported row.

You use the table RCV_HEADERS_INTERFACE only to create or update header information. The system requires that this table must by populated when you create an ASN, ASBN, or a receipt.

Errors

If the system detects errors, then the Receiving Open Interface populates the PO_INTERFACE_ERRORS table and the outbound Application Advice e-Commerce Gateway interface tables. A separate process in e-Commerce Gateway downloads the contents of the Outbound Application Advice Interface tables to the Outbound Application Advice flat file. For ASNs with billing information (also called ASBNs), if any lines are rejected, the Receiving Open Interface sets the INVOICE_STATUS_CODE to RCV_ASBN_NO_AUTO_INVOICE so that an invoice will not be created automatically from the rejected ASBN lines. You can view errors through the Receiving Interface Errors Report in Oracle Purchasing. To view errors specifically for ASBNs, use the Purchasing Interface Errors Report.

The system does not import rows that fail validation in the Receiving Open Interface tables. In other words, the system does not import rows that produce errors into Oracle Purchasing (into the RCV_SHIPMENT_HEADERS, RCV_SHIPMENTS_LINES, and other applicable Oracle Purchasing tables). For example, if an ASN contains multiple purchase orders, and if the purchase order number for one of the shipments is wrong, the shipment or the entire ASN fails, depending on how you set the *RCV: Fail All ASN Lines if One Line Fails* profile option.

Receiving Transaction Processor Activities

After performing header and line-level validation, the processor checks the profile option *RCV: Fail All ASN Lines if One Line Fails for an ASN/ASBN.* If the profile option is set to Yes and any line fails validation, then the processor fails the entire transaction. If the profile option is set to No (and the TEST_FLAG is not Y), the Receiving Transaction Processor performs the same steps that occur when you normally save receipt information in Oracle Purchasing for all successfully processed records. For all transactions other than ASN/ASBN, if you want a The Receiving Transaction Processor:

- Populates the RCV_SHIPMENT_HEADERS table in Oracle Purchasing with the receipt header information.
- Populates the RCV_SHIPMENT_LINES table in Oracle Purchasing for each receipt header entry in the RCV_SHIPMENT_HEADERS table in Oracle Purchasing.
- Populates the RCV_TRANSACTIONS table in Oracle Purchasing for each row in the RCV_SHIPMENT_HEADERS and RCV_SHIPMENT_LINES table if the column AUTO_TRANSACT_CODE in the RCV_TRANSACTIONS_INTERFACE table contains a value of RECEIVE or DELIVER.
- Updates supply for accepted line items in the MTL_SUPPLY and RCV_SUPPLY tables.
- Calls the Oracle Inventory module for processing DELIVER transactions.

- Calls the Oracle General Ledger module for processing financial transactions, such as receipt-based accruals.
- Updates the corresponding purchase orders with the final received and delivered quantities.

ASN Quantity Updates

While updating purchasing document quantities received, the Receiving Open Interface verifies that the quantity shipped was actually received for each item indicated on the ASN. If not, the interface populates the Application Advice history tables and the Application Advice e-Commerce Gateway interface tables with an error.

While updating the CUM quantity for Approved Supplier List items, the Receiving Open Interface also verifies that the new CUM quantity matches the supplier-specified CUM quantity. If not, the interface populates the Application Advice history tables and the Application Advice e-Commerce Gateway interface tables with an error. (CUM management is performed only if Oracle Supplier Scheduling is installed and CUM Management is enabled for the ship-to organization, the ASN item or items are defined in the Approved Supplier List, and the items are sourced from the supplier using a supply agreement blanket purchase order.)

Cascading Transaction Quantities for ASNs and Receipts

A purchase order sent to a supplier can include multiple lines and shipments. If the supplier does not provide a specific purchase order line number, release line number, or shipment number on the ASN but references a purchase order number, the Receiving Open Interface allocates the quantity on a first-in, first-out basis over all applicable purchase order and release shipments (if an item number is provided). To determine which shipment lines to consume, the Receiving Open Interface references all PO_LINE_LOCATIONS associated with the specified purchase order or blanket that have the same ship-to organization specified on the ASN. The order-by clause, NVL (PROMISED_DATE, NEED_BY_DATE, CREATION_DATE) determines the order in which quantities are consumed in a first-in, first-out basis. Therefore, multiple shipment lines matching the various purchase order shipment lines are created based on the allocation to the PO_LINE_LOCATIONS table, which stores lines corresponding to purchase order shipments.

The cascade works on a line-by-line basis, applying the remaining quantity to the last shipment line. *At the last line,* the Receiving Open Interface cascades up to the over-receipt tolerance. For example:

- There are 10 purchase order shipment lines of 100 units each, all with the same Need-By Date.
- In the Receiving Controls window in Oracle Purchasing, the Over Receipt Quantity

Tolerance is 10 percent meaning the Receiving Open Interface can consume 10 more units for the last shipment line if necessary.

- The actual ASN total quantity is 1,111, which exceeds your tolerance.

  If the Over Receipt Quantity Action code is set to Reject (and *RCV: Fail All ASN Lines if One Line Fails* is set to No), then Oracle Purchasing rejects the last ASN line (or the whole ASN if the ASN has only one line) and creates an error in the PO_INTERFACE_ERRORS table. Oracle Purchasing receives none of the units for those ASN lines that were rejected.

Oracle Purchasing does not require a Promised or Need-By date for an item that is unplanned; for unplanned items, Oracle Purchasing uses the CREATION_DATE in the order-by clause, NVL (PROMISED_DATE, NEED_BY_DATE, CREATION_DATE). If the cascade tries to allocate to an open shipment where the Receipt Date tolerance (the date after which a shipment cannot be received) is exceeded and the Receipt Date Action in the Receiving Controls window is set to Reject, Oracle Purchasing skips that shipment and goes to the next.

Setting Up the Receiving Open Interface

You must complete the following setup steps in Oracle Purchasing to use the Receiving Open Interface:

- Provide a Yes or No value for the profile option *RCV: Fail All ASN Lines if One Line Fails.* See Purchasing Profile Options, *Oracle Purchasing Users Guide*

- In the Receiving Options window in Oracle Purchasing, select Warning, Reject, or None in the ASN Control field to determine how Oracle Purchasing handles the receipt against a purchase order shipment for which an ASN exists. See Defining Receiving Options,*Oracle Purchasing User's Guide*

- If you are receiving ASNs in the Receiving Open Interface, install and set up e-Commerce Gateway. See *Oracle e-Commerce Gateway User's Guide*

All processing is initiated through standard report submission using the Submit Request window and choosing the Receiving Transaction Processor program. The concurrent manager manages; therefore you must ensure that processing has already been set up and that it is running.

Inserting into the Receiving Open Interface Table

You load receipt data from your source system or e-Commerce Gateway into the receiving headers and receiving transactions interface tables. For each row that you insert into the RCV_HEADERS_INTERFACE table, the Receiving Open Interface creates a shipment header; for each row that you insert into the RCV_TRANSACTIONS_INTERFACE table, the Receiving Open Interface creates one or

more shipment lines. You must provide values for all columns that are required. You may also have to provide values for columns that are conditionally required.

Required

You must specify values for columns in this category. The Receiving Open Interface requires values in these columns to process a receiving transaction whether the data is imported through the e-Commerce Gateway or through a program that you write. For example, HEADER_INTERFACE_ID is a required column; however, when receiving ASNs from suppliers through e-Commerce Gateway, e-Commerce Gateway provides the HEADER_INTERFACE_ID automatically. If a required value is not entered, the Receiving Open Interface inserts an error record in the PO_INTERFACE_ERRORS table.

Receiving Headers Interface Table Description

The following table contains information about the RVC_HEADERS_INTERFACE:

Column Name	Type	Required?	Receipt Type	Source Information
HEADER_INTERFACE_ID	Number	Yes	PO, ASN, REQ, INV, RMA	RCV_HEADERS_INTERFACE_S
GROUP_ID	Number	Yes	PO, ASN, REQ, INV, RMA	RCV_INTERFACE_GROUPS_S
PROCESSING_STATUS_CODE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	'PENDING'
RECEIPT_SOURCE_CODE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	PO/ASN='VENDOR' REQ='INTERNAL ORDER 'INV= 'INVENTORY' RMA= 'CUSTOMER'
TRANSACTION_TYPE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	Possible values are 'NEW', 'REPLACE', 'ADD', 'CANCEL'

Column Name	Type	Required?	Receipt Type	Source Information
LAST_UPDATE_DATE	Date	Yes	PO, ASN, REQ, INV, RMA	SYSDATE
LAST_UPDATED_BY	Number	Yes	PO, ASN, REQ, INV, RMA	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	Number	Yes	PO, ASN, REQ, INV, RMA	FND_GLOBAL.LOGIN_ID
CREATION_DATE	Date	Yes	PO, ASN, REQ, INV, RMA	SYSDATE
CREATED_BY	Number	Yes	PO, ASN, REQ, INV, RMA	FND_GLOBAL.USER_ID
VALIDATION_FLAG	Varchar2(1)	Yes	PO, ASN, REQ, INV, RMA	Yes
EDI_CONTROL_NUM	Varchar2(10)	No	ASN	
TEST_FLAG	Varchar2(1)	No		
NOTICE_CREATION_DATE	Date	No	ASN	
RECEIPT_NUM	Varchar2(30)	Conditionally	PO, ASN, REQ, INV, RMA	
RECEIPT_HEADER_ID	Number	Conditionally	ASN, Req, Inv	RCV_SHIPMENT_HEADERS.SHIPMENT_HEADER_ID
VENDOR_NAME	Varchar2(240)	Conditionally	PO, ASN	PO_VENDORS.VENDOR_NAME
VENDOR_NUM	Varchar2(20)	Conditionally	PO, ASN	PO_VENDORS.SEGMENT1

Column Name	Type	Required?	Receipt Type	Source Information
VENDOR_ID	Number	Conditionally	PO, ASN	PO_HEADERS_ALL.VENDOR_ID
VENDOR_SITE_CODE	Varchar2(35)	Conditionally	PO, ASN	PO_VENDOR_SITES_ALL.VENDOR_SITE_CODE
VENDOR_SITE_ID	Number	Conditionally	PO, ASN	PO_HEADERS_ALL.VENDOR_SITE_ID
FROM_ORGANIZATION_CODE	Varchar2(3)	Conditionally	Req, Inv	MTL_PARAMETERS.ORGANIZATION_CODE
FROM_ORGANIZATION_ID	Number	Conditionally	Req, Inv	RCV_SHIPMENT_HEADERS.ORGANIZATION_ID
SHIP_TO_ORGANIZATION_CODE	Varchar2(3)	Conditionally	PO, ASN, REQ, INV, RMA	MTL_PARAMETERS.ORGANIZATION_CODE
SHIP_TO_ORGANIZATION_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	PO/ASN= POLINE_LOCATIONS_ALL.SHIP_TO_ORGANIZATION_ID REQ/INV= RCV_SHIPMENT_LINES.TO_ORGANIZATION_ID RMA= OE_ORDER_LINES_ALL.SHIP_FROM_ORG_ID

Column Name	Type	Required?	Receipt Type	Source Information
LOCATION_CODE	Varchar2(60)	Conditionally	PO, ASN, REQ, INV, RMA	HR_LOCATIONS_ALL.LOCATION_CODE
LOCATION_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	PO/ASN= PO_LINE_LOCATIONS_ALL.SHIP_TO_LOCATION_ID REQ/INV= RCV_SHIPMENT_HEADERS.SHIP_TO_LOCATION_ID
BILL_OF_LADING	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
PACKING_SLIP	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
SHIPPED_DATE	Date	Optional	PO, ASN, REQ, INV, RMA	
FREIGHT_CARRIER_CODE	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
EXPECTED_RECEIPT_DATE	Date	Yes	PO, ASN, REQ, INV, RMA	
RECEIVER_ID	Number	Optional	PO, ASN, REQ, INV, RMA	
NUM_OF_CONTAINERS	Number	Optional	PO, ASN, REQ, INV, RMA	
WAYBILL_AIRBILL_NUM	Varchar2(20)	Optional	PO, ASN, REQ, INV, RMA	
COMMENTS	Varchar2(240)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
GROSS_WEIGHT	Number	Optional	PO, ASN, REQ, INV, RMA	
GROSS_WEIGHT_UOM_CODE	Varchar2(3)	Optional	PO, ASN, REQ, INV, RMA	
NET_WEIGHT	Number	Optional	PO, ASN, REQ, INV, RMA	
NET_WEIGHT_UOM_CODE	Varchar2(3)	Optional	PO, ASN, REQ, INV, RMA	
TAR_WEIGHT	Number	Optional	PO, ASN, REQ, INV, RMA	
TAR_WEIGHT_UOM_CODE	Varchar2(3)	Optional	PO, ASN, REQ, INV, RMA	
PACKAGING_CODE	Varchar2(5)	Optional	PO, ASN, REQ, INV, RMA	
CARRIER_METHOD	Varchar2(2)	Optional	PO, ASN, REQ, INV, RMA	
CARRIER_EQUIPMENT	Varchar2(10)	Optional	PO, ASN, REQ, INV, RMA	
SPECIAL_HANDLING_CODE	Varchar2(3)	Optional	PO, ASN, REQ, INV, RMA	
HAZARD_CODE	Varchar2(1)	Optional	PO, ASN, REQ, INV, RMA	
HAZARD_CLASS	Varchar2(4)	Optional	PO, ASN, REQ, INV, RMA	
HAZARD_DESCRIPTION	Varchar2(80)	Optional	PO, ASN, REQ, INV, RMA	
FREIGHT_TERMS	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
FREIGHT_BILL_NUMBER	Varchar2(35)	Optional	PO, ASN, REQ, INV, RMA	
INVOICE_NUM	Varchar2(30)	Conditionally	ASN	
INVOICE_DATE	Date	Conditionally	ASN	
TOTAL_INVOICE_AMOUNT	Number	Conditionally	ASN	
TAX_NAME	Varchar2(25)	Conditionally	ASN	
TAX_AMOUNT	Number	Conditionally	ASN	
FREIGHT_AMOUNT	Number	Optional	ASN	
CURRENCY_CODE	Varchar2(15)	Optional	ASN	
CONVERSION_RATE	Number	Optional	ASN	
CONVERSION_RATE_TYPE	Varchar2(30)	Optional	ASN	
CONVERSION_RATE_DATE	Date	Optional	ASN	
PAYMENT_TERMS_NAME	Varchar2(50)	Optional	ASN	
PAYMENT_TERMS_ID	Number	Optional	ASN	
ATTRIBUTE_CATEGORY	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE1	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
ATTRIBUTE2	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE3	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE4	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE5	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE6	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE7	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE8	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE9	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE10	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE11	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE12	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE13	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE14	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE15	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
USSGL_TRANSACTION_CODE	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
EMPLOYEE_NAME	Varchar2(240)	Conditionally	PO, ASN, REQ, INV, RMA	HR_EMPLOYEES.FULL_NAME
EMPLOYEE_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	HR_EMPLOYEES.EMPLOYEE_ID
INVOICE_STATUS_CODE	Varchar2(25)	Optional	ASN	
PROCESSING_REQUEST_ID	Number	NULL	PO, ASN, REQ, INV, RMA	
CUSTOMER_ACCOUNT_NUMBER	Number	Conditionally	RMA	HZ_CUST_ACCOUNTS.ACCOUNT_NUMBER
CUSTOMER_ID	Number	Conditionally	RMA	HZ_CUST_ACCOUNTS.CUST_ACCOUNT_ID
CUSTOMER_SITE_ID	Number	Conditionally	RMA	OE_ORDER_LINES_ALL.SHIP_FROM_ORG_ID
CUSTOMER_PARTY_NAME	Varchar2(360)	Conditionally	RMA	HZ_PARTIES.PARTY_NAME
REMIT_TO_SITE_ID	Number	Optional	ASN	
AUTO_TRANSACT_CODE	Varchar2(25)	Conditionally	PO, ASN, REQ, INV, RMA	
SHIPMENT_NUM	Varchar2(30)	Conditionally	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
ASN_TYPE	Varchar2(25)	Conditionally	Possible Values are ASN, ASBN, STD	

HEADER_INTERFACE_ID

The interface EDI header Sequence generated unique identifier.

GROUP_ID

The interface sequence generated group identifier for set processing. The GROUP_ID cannot be longer than nine digits.

PROCESSING_STATUS_CODE

This column indicates the status of each row in the RCV_HEADERS_INTERFACE table. The Receiving Open Interface selects a row for processing only when the value in this column is PENDING.

RECEIPT_SOURCE_CODE

This column indicates the source of the shipment. It tells the Receiving Open Interface whether the shipment is from an external supplier or an internal organization. Currently, this column can accept a value of onlyVENDOR.

TRANSACTION_TYPE

This column indicates the transaction purpose code for the shipment header. This column accepts a value of NEW or CANCEL.

LAST_UPDATE_DATE, LAST_UPDATED_BY, CREATION_DATE, and CREATED_BY

LAST_UPDATE_DATE indicates the date that the header record was last created or updated. LAST_UPDATED_BY indicates the loading program or user name identifier (ID) that was used to import the header record. CREATION_DATE indicates the date that the header record was created. CREATED_BY indicates the loading program or user ID that was used to import the header record. If you are importing data through e-Commerce Gateway, values are provided in these columns automatically.

VALIDATION_FLAG

This column indicates whether to validate a row before processing it. This column accepts values of Y or N. The Receiving Open Interface provides a default value of Y.

EDI_CONTROL_NUM

This is the EDI transaction control number if you are sending data via EDI.

TEST_FLAG

This flag indicates that the transaction is in test mode.

NOTICE_CREATION_DATE

This is the EDI transaction creation date and time at source if you are importing data via EDI.

RECEIPT_NUM

This column indicates the receipt number from the supplier. You must provide a value in this column if AUTO_TRANSACT_CODE is not SHIP, the TRANSACTION_TYPE or AUTO_TRANSACT_CODE in the RCV_TRANSACTIONS_INTERFACE table is not SHIP, and the Receipt Number Options Entry method (in the Receiving Options window) is Manual. The value in this column must be unique from the supplier for a period of one year.

RECEIPT_HEADER_ID

This is the receipt system ID.

VENDOR_NAME, VENDOR_NUM, or VENDOR_ID

These are required for PO, ASN, or ASBN-related transactions. Leave these columns blank when you receive against an RMA or intransit shipment.

VENDOR_NAME and VENDOR_NUM indicate the supplier name and number for the shipment. Both must be a valid name or number in Oracle Purchasing. Either one must be specified. (If you specify one, the Receiving Open Interface can derive the other.)

VENDOR_ID can be derived if either a VENDOR_NAME or VENDOR_NUM is provided. If no VENDOR_NAME or VENDOR_NUM is provided, you must provide a VENDOR_ID.

VENDOR_SITE_CODE

This is the supplier site code from PO/ASN; it is used to the derive VENDOR_SITE_ID.

VENDOR_SITE_ID

This is the source supplier site unique identifier.

FROM_ORGANIZATION_CODE

This is the source organization code; it is used to derive from the ORGANIZATION_ID.

FROM_ORGANIZATION_ID

This is the source organization unique identifier (internal transfers only).

SHIP_TO_ORGANIZATION_CODE or SHIP_TO_ORGANIZATION_ID

These columns indicate the destination organization for the shipment. A valid inventory organization code in Oracle Purchasing is required for an ASN. If the supplier does not know the ship-to organization, then the supplier can provide a ship-to location (SHIP_TO_LOCATION_CODE or SHIP_TO_LOCATION_ID) that is tied to an inventory organization in the Locations window, and the Receiving Open Interface can derive the inventory organization that way. A SHIP_TO_ORGANIZATION_CODE or SHIP_TO_ORGANIZATION_ID can be specified here in the RCV_HEADERS_INTERFACE table, at the header level, or in the RCV_TRANSACTIONS_INTERFACE table at the transaction line level. If the SHIP_TO_ORGANIZATION_CODE or SHIP_TO_ORGANIZATION_ID is specified at the header level, then it must apply to all shipments on the ASN. If it is specified at the line level, then it can be different for each line.

A SHIP_TO_ORGANIZATION_CODE or SHIP_TO_ORGANIZATION_ID enables the Receiving Open Interface to validate information at the line level before cascading quantities at the shipment level. This information helps the Receiving Open Interface determine if the supplier is providing valid item and shipment information.

LOCATION_CODE

This is the ship to and receiving location code.

LOCATION_ID

This is the receiving location unique identifier.

BILL_OF_LADING

This is the bill of lading number.

PACKING_SLIP

This is the packing slip number.

SHIPPED_DATE

This column indicates that the date the shipment was shipped. The value in this column is required for an ASN_TYPE of ASN or ASBN (for an ASN with billing information), and must be earlier than or equal to the system date. The value must also be earlier than or equal to the EXPECTED_RECEIPT_DATE.

FREIGHT_CARRRIER_CODE

This is the carrier who is responsible for shipment.

EXPECTED_RECEIPT_DATE

This is the expected arrival date of the shipment.

RECEIVER_ID

This is the employee unique identifier.

NUM_OF_CONTAINERS

This column indicates the umber of containers in the shipment.

WAYBILL_AIRBILL_NUM

This is the waybill or airbill number.

COMMENTS

These are the receiver's comments.

GROSS_WEIGHT and GROSS_WEIGHT_UOM_CODE

These columns indicate the shipment gross weight and shipment gross weight unit of measure.

NET_WEIGHT and NET_WEIGHT_UOM_CODE

These columns indicate the shipment net weight and shipment net weight unit of measure.

TAR_WEIGHT and TAR_WEIGHT_UOM_CODE

These columns indicate the shipment tar (container) weight and shipment tar (container) weight unit of measure.

PACKAGING_CODE

This is the shipment packing code.

CARRIER_METHOD

This is the carrier transportation method code.

CARRIER_EQUIPMENT

This is the carrier equipment description code.

SPECIAL_HANDLING_CODE

This is the special handling code.

HAZARD_CODE, HAZARD_CLASS, and HAZARD_DESCRIPTION

These are the hazardous material qualifier code of the shipment, hazardous material class of the shipment, and hazardous material description.

FREIGHT_TERMS

This is the freight payment method for example, prepaid or collect.

FREIGHT_BILL_NUMBER

This is the freight bill (PRO invoice) number.

INVOICE_NUM

A value for this column is required for ASBN transactions (if the ASN_TYPE is ASBN for an ASN with billing information). The value must be unique for the given supplier.

INVOICE_DATE

An invoice date is required for an ASBN transaction (if the ASN_TYPE is ASBN for an ASN with billing information).

TOTAL_INVOICE_AMOUNT

This column is required for ASBN transactions (ASNs with billing information). For ASBN transactions, you must provide a non-negative value in this column, even if that value is 0.

TAX_NAME and TAX_AMOUNT

EDI transaction 857 tax name associated with the shipment or billing notice, and the tax amount associated indicated at the header level.

FREIGHT_AMOUNT

This is the freight bill amount associated with the shipment or billing notice.

CURRENCY_CODE

This is the currency code associated with the shipment or billing notice.

CONVERSION_RATE_TYPE, CONVERSION_RATE, and CONVERSION_RATE_DATE

These are the exchange rate type, exchange rate, and the exchange rate date.

PAYMENT_TERMS_NAME and PAYMENT_TERMS_ID

These are the payment terms name and the payment terms unique identifier.

ATTRIBUTE_CATEGORY, ATTRIBUTE1, ATTRIBUTE2, ATTRIBUTE3, ATTRIBUTE4, ATRRIBUTE5, ATTRIBUTE6, ATTRIBUTE7, ATTRIBUTE8, ATTRIBUTE9, ATTRIBUTE10, ATTRIBUTE11, ATTRIBUTE12, ATTRIBUTE13, ATTRIBUTE13, ATTRIBUTE 14, and ATTRIBUTE15

These are descriptive flexfield segments.

USGGL_TRANSACTION_CODE

This is the United States standard general ledger transaction code.

EMPLOYEE_NAME or EMPLOYEE_ID

This column indicates the employee who created the shipment. You must provide a value in one of these columns if no value is provided in the corresponding columns in the RCV_TRANSACTIONS_INTERFACE table and if the AUTO_TRANSACT_CODE is RECEIVE. The value must be a valid employee name in Oracle Purchasing or Oracle applications.

INVOICE_STATUS_CODE

For ASN with billing information (ASBN) only, this code indicates when line items are rejected.

PROCESS_REQUEST_ID

This is the unique identifier for the request.

CUSTOMER_ACCOUNT_NUMBER and CUSTOMER_PARTY_NAME

This is the customer account number and customer name.

CUSTOMER_ID

This is the unique customer identifier. If this field is not populated, then the customer account number and customer name are required.

CUSTOMER_SITE_ID

This is the unique identifier for the customer site.

REMIT_TO_SITE_ID

This is the remit-to site identifier.

AUTO_TRANSACT_CODE

This column accepts values of SHIP, RECEIVE, or DELIVER. A value is required for ASN (ASN_TYPE) transactions. The value should be RECEIVE if you want to do a receiving transaction and if you provide an EMPLOYEE_NAME or EMPLOYEE_ID at the header level.

SHIPMENT_NUM

This column indicates the shipment number from the supplier. If no value is provided in this column, the Receiving Open Interface tries to default a value from the

PACKING_SLIP or INVOICE_NUM columns. The value of the SHIPMENT_NUM column must be unique from the SUPPLIER, SUPPLIER SITE for a period of one year, and validation on SHIPMENT_NUM happens only when a unique supplier site could be derived for the supplier (when supplier site is left NULL in the header interface table).

ASN_TYPE

This column accepts values of ASN or ASBN to indicate whether the transaction is for an ASN or an ASN with billing information. A value is required only when importing ASNs or ASBNs through e-Commerce Gateway. Leaving this column blank means that the transaction is not for an ASN or ASBN, but for a receipt, depending on the values in the AUTO_TRANSACT_CODE and TRANSACTION_TYPE columns.

Receiving Transactions Interface Table Description

The following table contains information about the RVC_TRANSACTIONS_INTERFACE:

Column Name	Type	Required?	Receipt Type	Source Information
INTERFACE_TRANSACTION_ID	Number	Yes	PO, ASN, REQ, INV, RMA	RCV_TRANSACTIONS_INTERFACE_S
GROUP_ID	Number	Yes	PO, ASN, REQ, INV, RMA	RCV_INTERFACE_GROUPS_S
LAST_UPDATE_DATE	Date	Yes	PO, ASN, REQ, INV, RMA	SYSDATE
LAST_UPDATED_BY	Number	Yes	PO, ASN, REQ, INV, RMA	FND_GLOBAL.USER_ID
CREATION_DATE	Date	Yes	PO, ASN, REQ, INV, RMA	SYSDATE
CREATED_BY	Number	Yes	PO, ASN, REQ, INV, RMA	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	Number	Yes	PO, ASN, REQ, INV, RMA	FND_GLOBAL.LOGIN_ID

Column Name	Type	Required?	Receipt Type	Source Information
TRANSACTION_TYPE	Varchar2	Yes	PO, ASN, REQ, INV, RMA	Possible values: SHIP, RECEIVE, DELIVER, TRANSFER, ACCEPT, REJECT, CORRECT, RETURN TO VENDOR, RETURN TO RECEIVING, RETURN TO CUSTOMER
TRANSACTION_DATE	Date	Yes	PO, ASN, REQ, INV, RMA	
PROCESSING_STATUS_CODE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	PENDING
PROCESSING_MODE_CODE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	Possible values: BATCH, IMMEDIATE, ONLINE
QUANTITY	Number	Yes	PO, ASN, REQ, INV, RMA	
UNIT_OF_MEASURE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	MTL_ITEM_UOMS_VIEW.UNIT_OF_MEASURE
RECEIPT_SOURCE_CODE	Varchar2(25	Yes	PO, ASN, REQ, INV, RMA	PO/ASN: VENDOR REQ: INTERNAL ORDER INV: INVENTORY RMA: CUSTOMER

Column Name	Type	Required?	Receipt Type	Source Information
SOURCE_DOCUMENT_CODE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	Possible Values: PO, REQ, INVENTORY, RMA
PARENT_TRANSACTION_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	RCV_TRANSACTIONS.TRANSACTION_ID
DESTINATION_TYPE_CODE	Varchar2(25)	Yes	PO, ASN, REQ, INV, RMA	Possible Values: 'RECEIVING', 'INVENTORY', 'EXPENSE', 'SHOP FLOOR'
ITEM_NUM	Varchar2(81)	Conditionally	PO, ASN, REQ, INV, RMA	
DOCUMENT_NUM	Varchar2(30)	Yes	PO, ASN, REQ, INV, RMA	
DOCUMENT_LINE_NUM	Number	Yes	PO, ASN, REQ, INV, RMA	
DOCUMENT_SHIPMENT_LINE_NUM	Number	Yes	PO, ASN, REQ, INV, RMA	
DOCUMENT_DISTRIBUTION_NUM	Number	Yes	PO, ASN, REQ, INV, RMA	
VALIDATION_FLAG	Varchar2(1)	Yes	PO, ASN, REQ, INV, RMA	Y
PARENT_INTERFACE_TXN_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID

Column Name	Type	Required?	Receipt Type	Source Information
REQUEST_ID	Number	Null	PO, ASN, REQ, INV, RMA	
PROGRAM_APPLICATION_ID	Number	Null	PO, ASN, REQ, INV, RMA	
PROGRAM_ID	Number	Null	PO, ASN, REQ, INV, RMA	
PROGRAM_UPDATE_DATE	Date	Null	PO, ASN, REQ, INV, RMA	
PROCESSING_REQUEST_ID	Number	Null	PO, ASN, REQ, INV, RMA	
CATEGORY_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	
INTERFACE_SOURCE_CODE	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
INTERFACE_SOURCE_LINE_ID	Number	Null		
INV_TRANSACTION_ID	Number	Null		
ITEM_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID
ITEM_DESCRIPTION	Varchar2(240)	Conditionally	PO, ASN, REQ, INV, RMA	
ITEM_REVISION	Varchar2(3)	Conditionally	PO, ASN, REQ, INV, RMA	
UOM_CODE	Varchar2(3)	Yes	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
EMPLOYEE_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	HR_EMPLOYEES.EMPLOYEE_ID
SHIPMENT_HEADER_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	RCV_SHIPMENT_HEADERS.SHIPMENT_HEADER_ID
SHIPMENT_LINE_ID	Number	Conditionally	ASN, REQ, INV	RCV_SHIPMENT_HEADERS.SHIPMENT_LINE_ID
SHIP_TO_LOCATION_ID	Number	Conditionally		PO/ASN: PO_LINE_LOCATIONS_ALL.SHIP_TO_LOCATION_ID REQ/INV: RCV_SHIPMENT_HEADERS.SHIP_TO_LOCATION_ID
PRIMARY_QUANTITY	Number	Conditionally		
PRIMARY_UNIT_OF_MEASURE	Varchar2(25)	Conditionally		MTL_SYSTEM_ITEMS_KFV.PRIMARY_UNIT_OF_MEASURE
VENDOR_ID	Number	Conditionally	PO, ASN	PO_HEADERS_ALL.VENDOR_ID
VENDOR_SITE_ID	Number	Conditionally	PO, ASN	PO_HEADERS_ALL.VENDOR_SITE_ID

Column Name	Type	Required?	Receipt Type	Source Information
FROM_ORGANIZATION_ID	Number	Conditionally	INV, RMA	RCV_SHIPMENT_HEADERS.ORGANIZATION_ID
FROM_SUBINVENTORY	Varchar2(10)	Optional	INV, RMA	
TO_ORGANIZATION_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	
INTRANSIT_OWNING_ORG_ID	Number	Optional	REQ, INV	
ROUTING_HEADER_ID	Number	Optional	PO, ASN, REQ, INV, RMA	
ROUTING_STEP_ID	Number	Null		
PO_HEADER_ID	Number	Conditionally	PO, ASN	PO_HEADERS_ALL.PO_HEADER_ID
PO_REVISION_NUM	Number	Conditionally	PO, ASN	
PO_RELEASE_ID	Number	Conditionally	PO, ASN	PO_RELEASES_ALL.PO_RELEASE_ID
PO_LINE_ID	Number	Conditionally	PO, ASN	PO_LINES_ALL.PO_LINE_ID
PO_LINE_LOCATION_ID	Number	Conditionally	PO, ASN	PO_LINE_LOCATIONS_ALL.LINE_LOCATION_ID
PO_UNIT_PRICE	Number	Conditionally	Conditionally	

Column Name	Type	Required?	Receipt Type	Source Information
CURRENCY_CODE	Varchar2(15)	Optional	PO, ASN, REQ, INV, RMA	
CURRENCY_CONVERSION_TYPE	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
CURRENCY_CONVERSION_RATE	Number	Optional	PO, ASN, REQ, INV, RMA	
CURRENCY_CONVERSION_DATE	Date	Optional	PO, ASN, REQ, INV, RMA	
PO_DISTRIBUTION_ID	Number	Conditionally	PO, ASN	PO_DISTRIBUTIONS_ALL.DISTRIBUTION_ID
REQUISITION_LINE_ID	Number	Null	REQ	PO_REQUISITION_LINES_ALL.REQUISITION_LINE_ID
REQ_DISTRIBUTION_ID	Number	Null	REQ	PO_REQ_DISTRIBUTIONS_ALL.DISTRIBUTION_ID
CHARGE_ACCOUNT_ID	Number	Optional	PO, ASN, REQ	
SUBSTITUTE_UNORDERED_CODE	Varchar2(25)	Optional	PO, ASN	
RECEIPT_EXCEPTION_FLAG	Varchar2(1)	Optional	PO, ASN, REQ, INV, RMA	
ACCRUAL_STATUS_CODE	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
INSPECTION_STATUS_CODE	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
INSPECTION_QUALITY_CODE	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
DELIVER_TO_PERSON_ID	Number	Optional	PO, ASN ,REQ	
LOCATION_ID	Number	Optional	PO, ASN, REQ, INV, RMA	
DELIVER_TO_LOCATION_ID	Number	Optional	PO, ASN, REQ, INV, RMA	
SUBINVENTORY	Varchar2(10)	Conditionally	PO, ASN, REQ, INV, RMA	
LOCATOR_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	
WIP_ENTITY_ID	Number	Optional		
WIP_LINE_ID	Number	Optional		
DEPARTMENT_CODE	Varchar2(10)	Optional		
WIP_REPETITIVE_SCHEDULE_ID	Number	Optional		
WIP_OPERATION_SEQ_NUM	Number	Optional		
WIP_RESOURCE_SEQ_NUM	Number	Optional		
BOM_RESOURCE_ID	Number	Optional		

Column Name	Type	Required?	Receipt Type	Source Information
SHIPMENT_NUM	Varchar2(30)	Conditionally	PO, ASN, REQ, INV, RMA	RCV_SHIPMENT_HEADERS.SHIPMENT_NUM
FREIGHT_CARRIER_CODE	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
BILL_OF_LADING	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
PACKING_SLIP	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
SHIPPED_DATE	Date	Optional		
EXPECTED_RECEIPT_DATE	Date	Conditionally	PO, ASN, REQ, INV, RMA	
ACTUAL_COST	Number	Optional		
TRANSFER_COST	Number	Optional		
TRANSPORTATION_COST	Number	Optional		
TRANSPORTATION_ACCOUNT_ID	Number	Optional		
NUM_OF_CONTAINERS	Number	Optional	PO, ASN, REQ, INV, RMA	
WAYBILL_AIRBILL_NUM	Varchar2(20)	Optional		
VENDOR_ITEM_NUM	Varchar2(25)	Optional	PO, ASN	
VENDOR_LOT_NUM	Varchar2(30)	Optional	PO, ASN	

Column Name	Type	Required?	Receipt Type	Source Information
RMA_REFERENCE	Varchar2(30)	Optional	PO, ASN	
COMMENTS	Varchar2(24)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE_CATEGORY	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE1	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE2	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE3	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE4	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE5	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE6	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE7	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE8	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE9	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE10	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE11	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
ATTRIBUTE12	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE13	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE14	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
ATTRIBUTE15	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE_CATEGORY	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE1	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE2	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE3	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE4	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE5	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE6	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE7	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE8	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
SHIP_HEAD_ATTRIBUTE9	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE10	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE11	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE12	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE13	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE14	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_HEAD_ATTRIBUTE15	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE_CATEGORY	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE1	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE2	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE3	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE4	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE5	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
SHIP_LINE_ATTRIBUTE6	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE7	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE8	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE9	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE10	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE11	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE12	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE13	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE14	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_LINE_ATTRIBUTE15	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
USSGL_TRANSACTION_CODE	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
GOVERNMENT_CONTEXT	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
REASON_ID	Number	Optional	PO, ASN, REQ, INV, RMA	
DESTINATION_CONTEXT	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
SOURCE_DOC_QUANTITY	Number	Optional	PO, ASN, REQ, INV, RMA	
SOURCE_DOC_UNIT_OF_MEASURE	Varchar2(25)	Optional	PO, ASN, REQ, INV, RMA	
MOVEMENT_ID	Number	Null		
HEADER_INTERFACE_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	
VENDOR_CUM_SHIPPED_QUANTITY	Number	Null		
TRUCK_NUM	Varchar2(35)	Optional	PO, ASN, REQ, INV, RMA	
SHIP_TO_LOCATION_CODE	Varchar2(60)	Conditionally	PO, ASN, REQ, INV, RMA	MTL_PARAMETERS.ORGANIZATION_CODE
CONTAINER_NUM	Varchar2(35)	Optional		
SUBSTITUTE_ITEM_NUM	Varchar2(81)	Optional	PO, ASN	
NOTICE_UNIT_PRICE	Number	Optional	ASN	
ITEM_CATEGORY	Varchar2(81)	Conditionally	PO, ASN, REQ, INV, RMA	
LOCATION_CODE	Varchar2(60)	Conditionally	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
VENDOR_NAME	Varchar2(240)	Conditionally	PO, ASN	PO_VENDORS.VENDOR_NAME
VENDOR_NUM	Varchar2(30)	Conditionally	PO, ASN	PO_VENDORS.SEGMENT1
VENDOR_SITE_CODE	Varchar2(15)	Conditionally	PO, ASN	PO_VENDOR_SITES_ALL.VENDOR_SITE_CODE
FROM_ORGANIZATION_CODE	Varchar2(3)	Null	REQ, INV	RCV_SHIPMENT_HEADERS.ORGANIZATION_ID
TO_ORGANIZATION_CODE	Varchar2(3)	Null	PO, ASN, REQ, INV, RMA	MTL_PARAMETERS.ORGANIZATION_CODE
INTRANSIT_OWNING_ORG_CODE	Varchar2(3)	Optional	INV	
ROUTING_CODE	Varchar2(30)	Optional	PO, ASN, REQ, INV, RMA	
ROUTING_STEP	Varchar2(30)	Null		
RELEASE_NUM	Number	Conditionally	PO, ASN	PO_RELEASES_ALL.RELEASE_NUM
DELIVER_TO_PERSON_NAME	Varchar2(240)	Optional	PO, ASN, REQ, INV, RMA	
DELIVER_TO_LOCATION_CODE	Varchar2(60)	Optional	PO, ASN, REQ, INV, RMA	
USE_MTL_LOT	Number	Optional		

Column Name	Type	Required?	Receipt Type	Source Information
USE_MTL_SERIAL	Number	Optional		
LOCATOR	Varchar2(81)	Conditionally	PO, ASN, REQ, INV, RMA	
REASON_NAME	Varchar2(30)	Optional		
SUBSTITUTE_ITEM_ID	Number	Conditionally		
QUANTITY_SHIPPED	Number	Conditionally		
QUANTITY_INVOICED	Number	Conditionally		
TAX_NAME	Varchar2(15)	Conditionally		
TAX_AMOUNT	Number	Conditionally		
REQ_NUM	Varchar2(25)	Conditionally	REQ	PO_REQUISITIONS_HEADERS_ALL.SEGMENT1
REQ_LINE_NUM	Number	Conditionally	REQ	PO_REQUISITION_LINES_ALL.LINE_NUM
REQ_DISTRIBUTION_NUM	Number	Conditionally	REQ	PO_REQ_DISTRIBUTIONS_ALL.DISTRIBUTION_NUM
WIP_ENTITY_NAME	Varchar2(24)	Optional	ASN	
WIP_LINE_CODE	Varchar2(10)	Optional	ASN	

Column Name	Type	Required?	Receipt Type	Source Information
RESOURCE_CODE	Varchar2(30)	Optional	ASN	
SHIPMENT_LINE_STATUS_CODE	Varchar2(25)	Optional		
BARCODE_LABEL	Varchar2(35)	Optional		
TRANSFER_PERCENTAGE	Number	Optional		
QA_COLLECTION_ID	Number	Conditionally		
COUNTRY_OF_ORIGIN_CODE	Varchar2(2)			
OE_ORDER_HEADER_ID	Number	Conditionally	RMA	OE_ORDER_HEADERS_ALL.HEADER_ID
OE_ORDER_LINE_ID	Number	Conditionally	RMA	OE_ORDER_LINES_ALL.LINE_ID
CUSTOMER_ID	Number	Conditionally	RMA	HZ_CUST_ACCOUNTS.CUST_ACCOUNT_ID
CUSTOMER_SITE_ID	Number	Conditionally	RMA	OE_ORDER_LINES_ALL.SHIP_FROM_ORG_ID
CUSTOMER_ITEM_NUM	Varchar2(50)	Conditionally	RMA	MTL_CUSTOMER_ITEMS.CUSTOMER_ITEM_NUMBER
CREATE_DEBIT_MEMO_FLAG	Varchar2(1)	Optional	PO, ASN	

Column Name	Type	Required?	Receipt Type	Source Information
PUT_AWAY_RULE_ID	Number	Null		
PUT_AWAY_STRATEGY_ID	Number	Null		
LPN_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	
TRANSFER_LPN_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	
COST_GROUP_ID	Number	Null		
MOBILE_TXN	Varchar2(2)	Null	Y for a row created through Oracle Warehouse Management mobile page. For Receiving Open Interface, this should be N.	
MMTT_TEMP_ID	Number	Null		
TRANSFER_COST_GROUP_ID	Number	Null		
SECONDARY_QUANTITY	Number	Conditionally	PO, ASN, REQ, INV, RMA	
SECONDARY_UNIT_OF_MEASURE	Varchar2(25)	Conditionally	PO, ASN, REQ, INV, RMA	
SECONDARY_UOM_CODE	Varchar2(3)	Conditionally	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
QC_GRADE	Varchar2(150)	Optional	PO, ASN, REQ, INV, RMA	
FROM_LOCATOR	Varchar2(81)	Conditionally	REQ, INV	
FROM_LOCATOR_ID	Number	Conditionally	REQ, INV	
PARENT_SOURCE_TRANSACTION_NUM	Varchar2(25)	Null		
INTERFACE_AVAILABLE_QTY	Number	Null		
INTERFACE_TRANSACTION_QTY	Number	Null		
INTERFACE_AVAILABLE_AMT	Number	Null		
INTERFACE_TRANSACTION_AMT	Number	Null		
LICENSE_PLATE_NUMBER	Varchar2(30)	Conditionally	PO, ASN, REQ, INV, RMA	
SOURCE_TRANSACTION_NUM	Varchar2(25)	Null		
TRANSFER_LICENSE_PLATE_NUMBER	Varchar2(30)	Conditionally		
LPN_GROUP_ID	Number	Conditionally	PO, ASN, REQ, INV, RMA	

Column Name	Type	Required?	Receipt Type	Source Information
ORDER_TRANSACTION_ID	Number	Null		
CUSTOMER_ACCOUNT_NUMBER	Number	Conditionally	RMA	HZ_CUST_ACCOUNTS.ACCOUNT_NUMBER
CUSTOMER_PARTY_NAME	Varchar2(360)	Conditionally	RMA	HZ_PARTIES.PARTY_NAME
OE_ORDER_LINE_NUM	Number	Conditionally	RMA	OE_ORDER_LINES_ALL.LINE_NUMBER
OE_ORDER_NUM	Number	Conditionally	RMA	OE_ORDER_HEADERS_ALL.ORDER_NUMBER
CUSTOMER_ITEM_ID	Number	Conditionally	RMA	MTL_CUSTOMER_ITEMS.CUSTOMER_ITEM_ID
AMOUNT	Number	Conditionally	PO, ASN	
JOB_ID	Number	Optional	PO	
TIMECARD_ID	Number	Optional	PO	
TIMECARD_OVN	Number	Optional	PO	
ERECORD_ID	Number	Null		
PROJECT_ID	Number	Null		
TASK_ID	Number	Null		
ASN_ATTACH_ID	Number	Null		

Column Name	Type	Required?	Receipt Type	Source Information
AUTO_TRANSACT_CODE	Varchar2(25)	Conditionally	PO, ASN, REQ, INV, RMA	Possible values: SHIP, RECEIVE, DELIVER

INTERFACE_TRANSACTION_ID

Oracle Purchasing provides a unique-sequence generator to generate a unique identifier for the receiving transaction line. If you are importing data through e-Commerce Gateway, a value is provided automatically.

GROUP_ID

Oracle Purchasing provides a group identifier for a set of transactions that should be processed together. The value in this column must match the GROUP_ID in the RCV_HEADERS_INTERFACE table. The GROUP_ID cannot be longer than nine digits.

LAST_UPDATE_DATE, LAST_UPDATED_BY, CREATION_DATE, LAST_UPDATE_LOGIN, and CREATED_BY

LAST_UPDATE_DATE indicates that the date the line was last created or updated. LAST_UPDATED_BY indicates the loading program or user name identifier (ID) that was used to import the line. CREATION_DATE indicates the date that the line was created. CREATED_BY indicates the loading program or user ID that was used to import the line. If you are importing data through e-Commerce Gateway, values are provided in these columns automatically.

TRANSACTION_TYPE

This column indicates the transaction purpose code. Valid values of TRANSACTION_TYPE are SHIP, RECEIVE, ACCEPT, REJECT, TRANSFER, CORRECT, DELIVER, RETURN TO RECEIVING, RETURN TO VENDOR, and RETURN TO CUSTOMER.

TRANSACTION_DATE

This column indicates the date of the transaction. The date must either be in an open or future entry Oracle Purchasing and Oracle General Ledger period. If Oracle Inventory is installed the date must be in an open or future Oracle Inventory period.

PROCESSING_STATUS_CODE

This column indicates the status of each row in the RCV_TRANSACTIONS_INTERFACE table. The Receiving Open Interface selects a row for processing only when the value in this column is PENDING.

PROCESSING_MODE_CODE

This column defines how the Receiving Open Interface is to be called. It accepts a value of BATCH only. You initiate one of these values when you submit the Receiving Transaction Processor program through the Submit Request window.

TRANSACTION_STATUS_CODE

This column indicates the status of the transaction record. The Receiving Open Interface provides a value of ERROR or COMPLETED.

During the cascade process, this quantity is allocated across all purchase order shipments in a first-in, first-out manner if the DOCUMENT_SHIPMENT_LINE_NUM is not specified. The cascade applies up to the amount ordered. However, if the quantity exceeds the quantity on the purchase order shipments, then the last purchase order shipment consumes the quantity ordered plus the allowable over-receipt tolerance.

All tolerances are checked as the quantity is cascaded. If the expected delivery date is not within the Receipt Date tolerance (the date after which a shipment cannot be received), and the Receipt Date Action in the Receiving Controls window is set to Reject, Purchasing skips the PO_LINE_LOCATIONS row and goes to the next.

QUANTITY, AMOUNT

You must specify either the quantity or the amount, depending on the line type being received against. For CORRECT transactions, it is possible for these to be negative. Negative corrections are limited to the available supply on the parent transaction.

UNIT_OF_MEASURE

This column indicates the shipment quantity unit of measure (UOM) and is not required when transacting against an amount-based line. If the UOM is different from the primary UOM defined in Oracle Purchasing or the source document UOM, then you must define a UOM conversion between the two UOMs.

RECEIPT_SOURCE_CODE

This column indicates the source of the shipment. Valid values of RECEIPT_SOURCE_CODE are VENDOR, CUSTOMER, INTERNAL ORDER, and INVENTORY. The Receiving Open Interface can derive the value here if one is

provided in the RCV_HEADERS_INTERFACE table.

SOURCE_DOCUMENT_CODE

This column indicates the document type for the shipment. Valid values of SOURCE_DOCUMENT_CODE are PO, RMA, REQ, and INVENTORY.

PARENT_TRANSACTION_ID

This column is the unique identifier of the parent receiving transaction.

DESTINATION_TYPE_CODE

You must provide a value for this column if the AUTO_TRANSACT_CODE is DELIVER. If you do not provide a value, the Receiving Open Interface uses the Destination Type on the purchase order shipment.

DOCUMENT_LINE_NUM, ITEM_NUM, VENDOR_ITEM_NUM, ITEM_ID, or PO_LINE_ID

You must provide a value for at least one of these columns, or for the CATEGORY_ID (or ITEM_CATEGORY) and ITEM_DESCRIPTION columns. If at least one value is provided, the Receiving Open Interface can derive some other values. For example, if a PO_LINE_ID is provided, the Receiving Open Interface can derive the ITEM_NUM and ITEM_ID.

DOCUMENT_LINE_NUM indicates the line number against which you are receiving. The value in this column must be a valid number for the purchase order that you are receiving against.

ITEM_NUM indicates the Purchasing item number of the item that you are receiving. The item number must be defined in Oracle Purchasing for the DOCUMENT_NUM provided and the SHIP_TO_ORGANIZATION_CODE.

VENDOR_ITEM_NUM indicates the supplier item number of the item that you are receiving. The value in this column must be defined in Oracle Purchasing as a supplier item number on the specified purchase order.

DOCUMENT_SHIPMENT_LINE_NUM and DOCUMENT_SHIPMENT_DISTRIBUTION_NUM

These columns are the purchase order shipment line number and distribution number.

VALIDATION_FLAG

This column tells the Receiving Open Interface whether to validate the row before processing it. It accepts values of Y or N. The Receiving Open Interface enters a default value of Y.

PARENT_INTERFACE_TXN_ID

This column is the parent receive transactions interface ID for the current row.

REQUEST_ID, PROGRAM_APPLICATION_ID, PROGRAM_ID, PROGRAM_UPDATE_DATE

These columns are standard who columns.

PROCESSING_REQUEST_ID

This column identifies the concurrent request that is processing the interface row. The Receiving Transaction processor sets this column.

ITEM_CATEGORY or CATEGORY_ID, or DOCUMENT_LINE_NUM or PO_LINE_ID

If you receive a shipment for an item that is not defined in Inventory (a one-time item), you must provide an ITEM_CATEGORY or CATEGORY_ID, or the DOCUMENT_LINE_NUM that the supplier is shipping against. This way, the Receiving Open Interface can match the line and allocate the quantity shipped. If you do not provide a value for ITEM_CATEGORY or CATEGORY_ID for a one-time item, you must provide a value for DOCUMENT_LINE_NUM or PO_LINE_ID.

ITEM_DESCRIPTION

This column indicates the item description. If no item description is provided, the Receiving Open Interface gets the item description from the purchase order line if a PO_LINE_ID or similar column is provided. See the next description.

ITEM_REVISION

You must provide a value if the item is under revision control and you have distributions with a destination type of Inventory. The value must be valid (defined in Oracle Purchasing) for the item you are receiving and the organization that you are receiving in. If no value is provided and one is required, the Receiving Open Interface defaults the latest implemented revision.

INTERFACE_SOURCE_LINE_CODE and INTERFACE_SOURCE_LINE_ID

These columns contain the source of the interface row and the source line identifier of the interface row.

UOM_CODE, PRIMARY_QUANTITY, PRIMARY_UNIT_OF_MEASURE

These columns are the transaction unit of measure code, primary unit of measure of the item, and the transaction quantity in the primary unit of measure for the item.

EMPLOYEE_ID

A value in this column is required if the TRANSACTION_TYPE is DELIVER. The value can be derived if an EMPLOYEE_NUM is provided in the RCV_HEADERS_INTERFACE table.

SHIPMENT_HEADER_ID and SHIPMENT_LINE_ID

These columns are the shipment header unique identifier and the shipment line unique identifier.

SHIP_TO_LOCATION_CODE or SHIP_TO_LOCATION_ID

If a SHIP_TO_LOCATION_CODE or SHIP_TO_LOCATION_ID, or SHIP_TO_ORGANIZATION_CODE or SHIP_TO_ORGANIZATION_ID is provided at the header level, in the RCV_HEADERS_INTERFACE table, the Receiving Open Interface can derive the SHIP_TO_LOCATION_CODE or SHIP_TO_LOCATION_ID at the line level, in the RCV_TRANSACTIONS_INTERFACE table.

A value is always required in the SHIP_TO_LOCATION_CODE or SHIP_TO_LOCATION_ID column for shipment (ASN or ASBN) transactions.

If the supplier does not provide ship-to organization information, then you must to tie your ship-to locations to a single Inventory organization in the Locations window. This way, the Receiving Open Interface can derive an organization based on the ship-to location.

VENDOR_NAME, VENDOR_NUM, or VENDOR_ID

At least one of these columns is required if they are not already provided in the RCV_HEADERS_INTERFACE table. These columns are relevant only to PO, ASN, or ASBN-related transactions. Leave these columns blank when receiving against an RMA or intransit shipment.

VENDOR_SITE_ID and VENDOR_SITE_CODE

These columns are the supplier site unique identifier and vendor site code.

FROM_ORGANIZATION_ID and FROM_SUBINVENTORY

These columns are the unique identifier for the source organization and source subinventory.

TO_ORGANIZATION_CODE or TO_ORGANIZATION_ID

You must provide a value for at least one of these columns. (The Receiving Open

Interface can derive the other.) However, if you provide a SHIP_TO_LOCATION_CODE or SHIP_TO_LOCATION_ID, and that location is tied to an Inventory organization in the Locations window, then the Receiving Open Interface can derive the TO_ORGANIZATION_CODE and TO_ORGANIZATION_ID.

The TO_ORGANIZATION_CODE indicates the destination ship-to organization code. For standard receipts, ASNs, or ASBNs the TO_ORGANIZATION_CODE must match the code in the receipt header.

INTRANSIT_OWNING_ORG_ID

This column is the organization that owns the items while they are intransit.

ROUTING_HEADER_ID

This column is the receiving routing unique identifier.

DOCUMENT_NUM or PO_HEADER_ID

This columns are required for only PO, ASN, or ASBN-related transactions. Leave these columns blank when receiving against an RMA or intransit shipment.

The column DOCUMENT_NUM indicates the purchase order document number against which to receive. The value in this column must be a valid purchasing document in Oracle Purchasing. If you provide a value in either the DOCUMENT_NUM or PO_HEADER_ID column, the other can be derived.

PO_REVISION_NUM and PO_RELEASE_ID

These columns are the PO revision number and the PO release identifier.

PO_LINE_LOCATION_ID and PO_UNIT_PRICE

These columns are the PO shipment identifier and the PO unit price at the time of receipt.

CURRENCY_CODE, CURRENCY_CONVERSION, CURRENCY_CONVERSION_RATE, CURRENCY_CONVERSION_DATE

These are the currency code, conversion type, conversion rate, and conversion date.

PO_DISTIRBUTION_ID

This is the PO distribution identifier.

REQUISTION_LINE_ID

This column is the requisition line identifier.

REQ_DISTRIBUTION_ID

This column is the requisition distribution identifier.

CHARGE_ACCOUNT_ID

This column is the charge account identifier.

SUBSTITUTE_UNORDERED_CODE

This code indicates if the receipt is a substitute item or is an unordered receipt.

RECEIPT_EXCEPTION_FLAG

This flag indicates that there is an exception to the receipt.

ACCRUAL_STATUS_CODE

This column indicates the accrual status of the receipt.

INSPECTION_STATUS_CODE

This column indicates the quality inspection status of the receipt.

INSPECTION_QUALITY_CODE

This column indicates the quality inspection result of the receipt.

DELIVER_TO_PERSON_ID or DELIVER_TO_PERSON_NAME, SUBINVENTORY, and LOCATOR or LOCATOR_ID

Values are required in these columns if the AUTO_TRANSACT_CODE is DELIVER and if the Receiving Open Interface cannot find the values in the purchase order itself. Additionally, LOCATOR or LOCATOR_ID is required if a Locator Control option is selected for the delivery transaction at the item level (in the Master Items or Organization Items windows), subinventory level (in the Subinventories window in Inventory), or organization level (in the Organization window).

WIP_ENTITY_ID, WIP_LINE_ID, DEPARTMENT_CODE, WIP_REPETITIVE_SCHEDULE_ID, WIP_OPERATION_SEQ_NUM, WIP_RESOURCE_SEQ_NUM, and BOM_RESOURCE_ID

These columns relate to WIP jobs and assemblies. They are:

- WIP_ENTITY_ID: WIP job or repetitive assembly identifier
- WIP_LINE_ID: WIP line identifier
- DEPARTMENT_CODE: WIP department code
- WIP_REPETITIVE_SCHEDULE_ID: WIP repetitive schedule identifier
- WIP_OPERATION_SEQ_NUM: WIP operation sequence number within a routing
- WIP_RESOURCE_SEQ_NUM: WIP resource sequence number
- BOM_RESOURCE_ID: BOM resource unique identifier

SHIPMENT_NUMBER

This column is the supplier or source organization shipment number.

FREIGHT_CARRRIER_CODE

This column is the carrier who is responsible for shipment.

BILL_OF_LADING

This column is the bill of lading number.

PACKING_SLIP

This column is the packing slip number.

SHIPPED_DATE

This column indicates the that date the shipment was shipped. The value in this column is required for an ASN_TYPE of ASN or ASBN (for an ASN with billing information). This value must be earlier than or equal to the system date. It must also be earlier than or equal to the EXPECTED_RECEIPT_DATE.

EXPECTED_RECEIPT_DATE

A value in this column is required if none is provided in the RCV_HEADERS_INTERFACE table. The date must fall within the receipt date

tolerance for the shipments with which the receipt is being matched.

ACTUAL_COST, TRANSFER_COST and _TRANSPORTATION_COST

These columns are the actual cost for an item, the cost of transferring items between organizations, and the costing of shipping items between organizations.

TRANSPORTATION_ACCOUNT_ID

This column is the account to be charged for transportation costs.

NUM_OF_CONTAINERS

This column is the number of containers in the shipment.

VENDOR_ITEM_NUM and VENDOR_LOT_NUM

These columns are the supplier item number and the supplier lot number.

RMA_REFERENCE

This column is the RMA reference number for return_to_supplier items.

COMMENTS

These are the receiver's comments.

ATTRIBUTE_CATEGORY, ATTRIBUTE1, ATTRIBUTE2, ATTRIBUTE3, ATTRIBUTE4, ATRRIBUTE5, ATTRIBUTE6, ATTRIBUTE7, ATTRIBUTE8, ATTRIBUTE9, ATTRIBUTE10, ATTRIBUTE11, ATTRIBUTE12, ATTRIBUTE13, ATTRIBUTE13, ATTRIBUTE 14, and ATTRIBUTE15

These columns are descriptive flexfield segments.

SHIP_HEAD_ATTRIBUTE_CATEGORY, SHIP_HEAD_ATTRIBUTE1, SHIP_HEAD_ATTRIBUTE2, SHIP_HEAD_ATTRIBUTE3, SHIP_HEAD_ATTRIBUTE4, SHIP_HEAD_ATTRIBUTE5, SHIP_HEAD_ATTRIBUTE6, SHIP_HEAD_ATTRIBUTE7, SHIP_HEAD_ATTRIBUTE8, SHIP_HEAD_ATTRIBUTE9, SHIP_HEAD_ATTRIBUTE10, SHIP_HEAD_ATTRIBUTE11, SHIP_HEAD_ATTRIBUTE12, SHIP_HEAD_ATTRIBUTE13, SHIP_HEAD_ATTRIBUTE14, SHIP_HEAD_ATTRIBUTE15

These columns are shipment header descriptive flexfield segments.

SHIP_LINE_ATTRIBUTE2, SHIP_LINE_ATTRIBUTE3, SHIP_LINE_ATTRIBUTE4, SHIP_LINE_ATTRIBUTE5, SHIP_LINE_ATTRIBUTE6, SHIP_LINE_ATTRIBUTE7, SHIP_LINE_ATTRIBUTE8, SHIP_LINE_ATTRIBUTE9, SHIP_LINE_ATTRIBUTE10, SHIP_LINE_ATTRIBUTE11, SHIP_LINE_ATTRIBUTE12, SHIP_LINE_ATTRIBUTE13, SHIP_LINE_ATTRIBUTE14, SHIP_LINE_ATTRIBUTE15

These columns are shipment line descriptive flexfield segments.

USGGL_TRANSACTION_CODE and GOVERNMENT_CONTEXTS

These are the United States standard general ledger transaction code and descriptive flexfield context.

REASON_ID

This column is the transaction reason identifier.

DESTINATION_CONTEXT

This column is the destination descriptive flexfield context.

SOURCE_DOC_QUANTITY and SOURCE_DOC_UNIT_OF_MEASURE

These columns are the source document unit of measure and transaction quantity.

HEADER_INTERFACE_ID

The HEADER_INTERFACE_ID (and a corresponding row in RCV_HEADERS_INTERFACE) are required for SHIP or RECEIVE transactions. Child transactions should not have a new header.

Oracle Purchasing provides a unique identifier for the corresponding header. The value in this column must match the HEADER_INTERFACE_ID in the RCV_HEADERS_INTERFACE table. If you are importing data through e-Commerce Gateway, a value is provided automatically.

MOVEMENT_ID

This is the movement identifier.

VENDOR_CUM_SHIPPED_QTY

This is the supplier cumulative shipped quantity specified on the advanced shipment notice.

TRUCK_NUM

This is the truck number.

CONTAINER_NUM

This is the container number.

SUBSTITUTE_ITEM_NUM

This is the buyer's substitute item number specified on the ASN.

NOTICE_UNIT_PRICE

This is the EDI transaction 857 unit price

FROM_ORGANIZATION_CODE and INTRANSIT_OWNING_ORG_CODE

These are the from organization code for interorganization transfers and the in-transit owning organization code for interorganization transfers.

ROUTING_CODE

This is the receiving routing name.

ROUTING_STEP

This is the upgrade step unique identifier.

RELEASE_NUMBER

This is the release number for the purchase order.

DELIVER_TO_LOCATION_CODE or DELIVER_TO_LOCATION_ID

A value is required in one of these columns if the AUTO_TRANSACT_CODE is DELIVER. If you do not provide a value, the Receiving Open Interface uses the deliver-to location on the purchase order shipment.

PARENT_TRANSACTION_ID, PARENT_INTERFACE_TXN_ID, SOURCE_TRANSACTION_NUM, PARENT_SOURCE_TRANSACTION_NUM

If the transaction is not a root transaction, (a SHIP or RECEIVE), then the Receiving Open Interface needs to be able to find the appropriate parent. If the parent transaction is in the interface tables, and has not yet been processed, PARENT_INTERFACE_TXN_ID should be populated with the

RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID of the parent. If the parent has been processed in an earlier batch, PARENT_TRANSACTION_ID should be populated with the RCV_TRANSACTIONS.TRANSACTION_ID of the parent. The Receiving Open Interface can also derive the appropriate value if SOURCE_TRANSACTION_NUM is populated on the parent, and PARENT_SOURCE_TRANSACTION_NUM is populated on the new transaction. This enables equivalent hierarchies to be fed in directly from other systems.

USE_MTL_LOT and USE_MTL_SERIAL

These columns are parameters used for lot and serial processing.

REASON_NAME

This is column is the transaction reason name.

SUBSTITUTE_ITEM_ID

This column is the substitute item unique identifier that the buyer uses.

QUANTITY_SHIPPED and QUANTITY_INVOICED

These columns represent the quantity shipped and the quantity invoiced.

TAX_NAME and TAX_AMOUNT

These columns are the EDI transaction 857 tax name associated with the shipment or billing notice, and the tax amount associated at the header level.

REQ_NUM, REQ_LINE_NUM, and REQ_DISTRIBUTION_NUM

These columns represent requisition number, requisition line number, and requisition distribution number.

SHIPMENT_LINES_STATUS_CODE

This column is the status code for the shipment line.

BARCODE_LABEL

Bar-coded detail label serial number if indicated on ASN for inner packaging of this item; to be matched for barcode-scanned receipt updating.

TRANSFER_PERCENTAGE

This column is the transfer percentage for costing.

QA_COLLECTION_ID

This column is the primary key for Oracle Quality results.

COUNTRY_OF_ORIGIN

This column is the code for the manufacturing country.

OE_ORDER_NUM, OE_ORDER_HEADER_ID, OE_ORDER_LINE_NUM, OE_ORDER_LINE_ID

The OE_ORDER_HEADER_ID and OE_ORDER_LINE_ID are required for transactions against an RMA. They can be derived from OE_ORDER_NUM and OE_ORDER_LINE_NUM, or DOCUMENT_NUM and DOCUMENT_LINE_NUM.

CUSTOMER_ID

This column is the Unique Customer identifier. If this field is not populated, then the customer account number and customer name are required.

CUSTOMER_SITE_ID

This column is the unique identifier for the customer site.

CUSTOMER_ITEM_NUM

This is the customer item number.

CREATE_DEBIT_MEMO_FLAG

This parameter determines if you need to create a debit memo.

LICENSE_PLATE_NUMBER, LPN_ID, TRANSFER_LICENSE_PLATE_NUMBER, TRANSFER_LPN_ID

Once items have been packed into an LPN, it is necessary to specify a from and to LPN on each subsequent transaction. Failure to do so will lead to implicit unpacking of the LPN. The TRANSFER_LICENSE_PLATE_NUMBER must be specified when packing items into an LPN that is being created in the same batch of transactions. If the LPN already exists, either the TRANSFER_LPN_ID or the TRANSFER_LICENSE_PLATE_NUMBER is acceptable. Similarly, the from LPN_ID can be derived from the LICENSE_PLATE_NUMBER, if the LPN already exists.

COST_GROUP_ID and TRANSFER_COST_GROUP_ID

These columns are the Oracle Warehouse Management cost group identifier and the transfer cost group identifier.

MMTT_TEMP_ID

This column references the original inventory transaction for Oracle Warehouse Management.

SECONDARY_UNIT_OF_MEASURE, SECONDARY_UOM_CODE, and SECONDARY_QUANTITY,

These columns are the transaction secondary unit of measure, the transaction secondary unit of measure code, and the transaction quantity in the secondary unit of measure.

QC_GRADE

This column is the quality grade of the received item.

PARENT_SOURCE_TRANSACTION_NUM

This column is the parent source transaction number.

INTERFACE_AVAILABLE_QTY

This column is the available quantity for the receiving transaction interface rows.

INTERFACE_TRANACTION_QTY

This column is the available transaction quantity.

INTERFACE_AVAILABLE_AMT

This column is the available amount for the children receiving transaction interface rows.

INTERFACE_TRANSACTION_AMT

This column is the available transaction quantity.

SOURCE_TRANSACTION_NUM

This column is the source transaction number.

LPN_GROUP_ID

The LPN_GROUP_ID is used to group multiple Receiving transactions together when they represent a single physical LPN-based transaction (for example, moving an LPN from one location to another). The grouped transactions will succeed or fail as a unit, so if one row fails validation, all grouped rows will be marked as errors. This column will be automatically derived in cases where RVCTP explodes the LPN contents into the component transactions.

ORDER_TRANSACTION_ID

This column is the order transaction identifier.

CUSTOMER_ACCOUNT_NUMBER and CUSTOMER_PARTY_NAME

These columns are the customer account number and customer name.

CUSTOMER_ITEM_ID

This column is the customer item identifier.

AMOUNT

This is the amount.

JOB_ID

This column is the job identifier for services.

TIMECARD_ID

This column is the timecard identifier.

TIMECARD_OVN

This column represents revisions to the timecard.

ERECORD_ID

This is the eRecord identifier.

PROJECT_ID and TASK_ID

These parameters represent the project and task ID for the timecard.

ASN_ATTACH_ID

This is the ASN attach ID.

AUTO_TRANSACT_CODE

The AUTO_TRANSACT_CODE is relevant only to SHIP or RECEIVE transactions and indicates the automatic transaction creation code of the shipment. It accepts values of RECEIVE for a standard receipt, DELIVER for a standard receipt and delivery transaction, and SHIP for a shipment (ASN or ASBN) transaction.

Whether you can perform a standard receipt (RECEIVE) or direct receipt (DELIVER) depends on the ROUTING_HEADER_ID in the PO_LINE_LOCATIONS table and the Purchasing profile option *RCV: Allow routing override*.

The AUTO_TRANSACT_CODE in the RCV_TRANSACTIONS_INTERFACE table overrides that in the RCV_HEADERS_INTERFACE table, if the two values differ.

The table following table shows the combinations of TRANSACTION_TYPE and AUTO_TRANSACT_CODE values that you can choose in the RCV_TRANSACTIONS_INTERFACE table to create an ASN or ASBN shipment header and shipment lines, a receiving transaction, or a receiving and delivery transaction.

TRANSACTION_TYPE	NULL	RECEIVE	DELIVER
SHIP	Shipment header and shipment lines created	Receiving transaction created	Receiving and delivery transaction created
RECEIVE	Receiving transaction created	Receiving transaction created	Receiving and delivery transaction created

Oracle Warehouse Management License Plate Number Interface Table Description

The following table contains information about the WMS_LPN_INTERFACE table: For additional information about the WMS_LPN_INTERFACE table see Oracle Warehouse Management LPN Interface Table Supplemental Information, *Oracle Manufacturing APIs and Open Interfaces Manual*.

Column Name	Type	Required?	Receipt Type	Source Information
LICENSE_PLATE_NUMBER	VARCHAR2 (30)	Yes	PO, ASN, REQ, INV RMA	RCV_TRANSACTIONS_INTERFACE.TRANSFER_LICENSE_PLATE_NUMBER
LAST_UPDATE_DATE	DATE	Yes	PO, ASN, REQ, INV RMA	SYSDATE
LAST_UPDATED_BY	NUMBER	Yes	PO, ASN, REQ, INV RMA	FND_GLOBAL.USER_ID
CREATION_DATE	DATE	Yes	PO, ASN, REQ, INV RMA	SYSDATE
CREATED_BY	NUMBER	Yes	PO, ASN, REQ, INV RMA	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	NUMBER	Yes	PO, ASN, REQ, INV RMA	FND_GLOBAL.LOGIN_ID
PARENT_LICENSE_PLATE_NUMBER	VARCHAR2 (30)	Conditionally	PO, ASN, REQ, INV RMA	
OUTERMOST_LPN	VARCHAR2 (30)	Yes	PO, ASN, REQ, INV RMA	
SOURCE_GROUP_ID	NUMBER	Yes	PO, ASN, REQ, INV RMA	RCV_TRANSACTIONS_INTERFACE.LPN_GROUP_ID
LPN_ID	NUMBER	Optional		
INVENTORY_ITEM_ID	NUMBER	Optional		
REQUEST_ID	NUMBER	Optional		

Column Name	Type	Required?	Receipt Type	Source Information
PROGRAM_APPLICATION_ID	NUMBER	Optional		
PROGRAM_ID	NUMBER	Optional		
PROGRAM_UPDATE_DATE	DATE	Optional		
REVISION	VARCHAR2 (3)	Optional		
LOT_NUMBER	VARCHAR2 (80)	Optional		
SERIAL_NUMBER	VARCHAR2 (30)	Optional		
ORGANIZATION_ID	NUMBER	Optional		
SUBINVENTORY_CODE	VARCHAR2 (10)	Optional		
LOCATOR_ID	NUMBER	Optional		
COST_GROUP_ID	NUMBER	Optional		
PARENT_LPN_ID	NUMBER	Optional		
OUTERMOST_LPN_ID	NUMBER	Optional		
GROSS_WEIGHT_UOM_CODE	VARCHAR2 (3)	Optional		
GROSS_WEIGHT	NUMBER	Optional		
CONTENT_VOLUME_UOM_CODE	VARCHAR2 (3)	Optional		

Column Name	Type	Required?	Receipt Type	Source Information
CONTENT_VOLUME	NUMBER	Optional		
TARE_WEIGHT_UOM_CODE	VARCHAR2 (3)	Optional		
TARE_WEIGHT	NUMBER	Optional		
STATUS_ID	NUMBER	Optional		
LPN_CONTEXT	NUMBER	Optional		
SEALED_STATUS	NUMBER	Optional		
LPN_REUSABILITY	NUMBER	Optional		
HOMOGENEOUS_CONTAINER	NUMBER	Optional		
ATTRIBUTE_CATEGORY	VARCHAR2 (30)	Optional		
ATTRIBUTE1	VARCHAR2 (150)	Optional		
ATTRIBUTE2	VARCHAR2 (150)	Optional		
ATTRIBUTE3	VARCHAR2 (150)	Optional		
ATTRIBUTE4	VARCHAR2 (150)	Optional		
ATTRIBUTE5	VARCHAR2 (150)	Optional		
ATTRIBUTE6	VARCHAR2 (150)	Optional		

Column Name	Type	Required?	Receipt Type	Source Information
ATTRIBUTE7	VARCHAR2 (150)	Optional		
ATTRIBUTE8	VARCHAR2 (150)	Optional		
ATTRIBUTE9	VARCHAR2 (150)	Optional		
ATTRIBUTE10	VARCHAR2 (150)	Optional		
ATTRIBUTE11	VARCHAR2 (150)	Optional		
ATTRIBUTE12	VARCHAR2 (150)	Optional		
ATTRIBUTE13	VARCHAR2 (150)	Optional		
ATTRIBUTE14	VARCHAR2 (150)	Optional		
ATTRIBUTE15	VARCHAR2 (150)	Optional		
SOURCE_TYPE_ID	NUMBER	Optional		
SOURCE_HEADER_ID	NUMBER	Optional		
SOURCE_LINE_ID	NUMBER	Optional		
SOURCE_LINE_DETAIL_ID	NUMBER	Optional		
SOURCE_NAME	VARCHAR2 (30)	Optional		

Column Name	Type	Required?	Receipt Type	Source Information
ORGANIZATION_CODE	VARCHAR2 (3)	Optional		

LICENSE_PLATE_NUMBER

This is the license plate number (LPN).

PARENT_LICENSE_PLATE_NUMBER

This is the LPN of the container in which the current LPN resides.

OUTERMOST_LPN

This is the outermost LPN.

SOURCE_GROUP_ID

This is the source group of the LPN. It joins to the group_id column for transactions. For Receiving Open Interface transactions this column joins to the LPN_GROUP_ID on RCV_TRANSACTIONS_INTERFACE.

LPN_ID

This is a unique identifier for each row in this table.

INVENTORY_ITEM_ID

This is the unique ID of an inventory item associated with an LPN (for example, a container item).

PROGRAM_UPDATE_DATE, PROGRAM_ID, REQUEST_ID, LAST_UPDATE_LOGIN, CREATED_BY, CREATION_DATE, LAST_UPDATED_BY, and LAST_UPDATE_DATE

These are standard who columns.

REVISION

This is the revision of the container item if it is under revision control.

LOT_NUMBER

This is the lot number if the container item is under lot control.

SERIAL_NUMBER

This is the serial number if the container item is under serial control.

ORGANIZATION_ID

This is the current organization identifier.

SUBINVENTORY_CODE

This is the current subinventory code.

LOCATOR_ID

This is the current locator if the container item is under locator control.

COST_GROUP_ID

This is the he cost group identifier.

PARENT_LPN_ID

This is the LPN ID of the container in which the current LPN resides.

OUTERMOST_LPN_ID

This is the outermost LPN ID.

GROSS_WEIGHT_UOM_CODE

This is the UOM code for the weight of the container and its contents.

GROSS_WEIGHT

This is the weight of the container body and its contents.

CONTENT_VOLUME_UOM_CODE

This is the UOM code for the container content volume.

CONTENT_VOLUME

This is the total volume of the container contents.

TARE_WEIGHT_UOM_CODE

This is the UOM code for the tare weight of the container.

TARE_WEIGHT

This is the tare weight for the container and its contents.

STATUS_ID

This is the current status of the container instance (LPN).

LPN_CONTEXT

This is the LPN context.

SEALED_STATUS

This column represents if the container is sealed or not.

LPN_REUSABILITY

This column determines if the LPN is resusable (1. Permanent, 2. Disposable).

HOMOGENEOUS_CONTAINER

This column indicates whether mixed items are allowed within the LPN. The default null indicates mixed items are allowed within the LPN.

ATTRIBUTE_CATEGORY, ATTRIBUTE1, ATTRIBUTE2, ATTRIBUTE3, ATTRIBUTE4, ATTRIBUTE5, ATTRIBUTE6, ATTRIBUTE7, ATTRIBUTE8, ATTRIBUTE9, ATTRIBUTE10, ATTRIBUTE11, ATTRIBUTE12, ATTRIBUTE13, ATTRIBUTE14, and ATTRIBUTE15

These are descriptive flexfield segment columns.

SOURCE_TYPE_ID

This is the source type of the LPN.

SOURCE_HEADER_ID

This is the source header ID of the LPN.

SOURCE_LINE_ID

This is the source line ID of the LPN.

SOURCE_LINE_DETAIL_ID

This is the source line detail ID of the LPN.

SOURCE_NAME

This is the source name of the LPN.

Derived Data

The Receiving Open Interface derives or defaults derived columns using logic similar to that used by the Receipts, Receiving Transactions, or Manage Shipments windows. Oracle Purchasing never overrides information that you provide in derived columns.

When a column exists in both the RCV_HEADERS_INTERFACE and RCV_TRANSACTIONS_INTERFACE tables, if you provide a value for the column in the RCV_HEADERS_INTERFACE table, the Receiving Open Interface can derive a value for the same column in the RCV_TRANSACTIONS_INTERFACE table. The LOCATION_CODE in the headers table and SHIP_TO_LOCATION_CODE in the transactions table are examples of this. In general, the Receiving Open Interface tries first to derive values in the RCV_TRANSACTIONS_INTERFACE table based on values in the RCV_HEADERS_INTERFACE table; then, if no corresponding values are there, it tries to derive them from the purchase order.

Some examples of derivation are, in the RCV_HEADERS_INTERFACE table. The RECEIPT_NUM is derived if the AUTO_TRANSACT_CODE is DELIVER or RECEIVE and, in the RCV_TRANSACTIONS_INTERFACE table, the DESTINATION_TYPE_CODE is derived if the TRANSACTION_TYPE is DELIVER.

Optional Data

Optional columns in the interface tables use the same rules as their corresponding fields in the Receipts, Receiving Transactions, and Manage Shipments windows in Oracle Purchasing. Here are some of these rules:

- RELEASE_NUM must be a valid release number for the purchasing document number provided and, if a release number is not provided, the Receiving Open Interface allocates the quantity across all open shipments for all releases.

- DOCUMENT_SHIPMENT_LINE_NUM must be a valid number for the line that you are receiving against if the line number (DOCUMENT_LINE_NUM) is provided. If a DOCUMENT_SHIPMENT_LINE_NUM is not provided, the Receiving Open Interface allocates the shipment quantity against the shipments in a first-in, first-out order based on the PROMISED_DATE or the NEED_BY_DATE in the Oracle Purchasing tables.

- SUBSTITUTE_ITEM_NUM - must be defined in Oracle Purchasing as a related item for an item on the provided DOCUMENT_NUM. The original item must allow substitute receipts, and the supplier must be enabled to send substitute items. The substitute item also must be enabled as an Oracle Purchasing item.

- REASON_NAME indicates the transaction reason, as defined in the Transaction Reasons window in Inventory.

Some other example information about optional data is, in the RCV_HEADERS_INTERFACE table, the EXPECTED_RECEIPT_DATE must be later than or equal to the SHIPPED_DATE, if a SHIPPED_DATE is given. Also, if Oracle Supplier Scheduling is installed and set up, and the value in the VENDOR_CUM_SHIPPED_QUANTITY column does not match what you have received, then your supplier is notified through an Application Advice (if you are receiving ASNs through e-Commerce Gateway).

Validation

Oracle Purchasing validates all required columns in the interface tables. For specific information on the data implied by these columns, see the *Oracle eTechnical Reference Manual* .

Other Validation

If a row in the interface tables fails validation for any reason, the program sets the PROCESSING_STATUS_CODE to ERROR and enters details about errors on that row into the PO_INTERFACE_ERRORS table.

In general, the same validations are performed in the Receiving Open Interface tables as are performed in the Receipts, Receiving Transactions, and Manage Shipments windows.

Debugging

Debugging enables you perform a test run of the Receiving Open Interface, see and fix the errors, and run the program again. The system writes the logs to the table FND_LOG_MESSAGES table. In addition, the system includes debug information in the Receiving Transaction Processor Request Log in the BATCH/IMMEDIATE mode. You must set the following profile options so that the system can write the log.

- RCV: Debug Mode: Yes
- FND: Debug Log Enabled: Yes
- FND: Debug Log Level: Statement
- FND: Debug Log Module: purchase order percent
- INV: Debug Trace: Yes
- INV: Debug File (including the complete path): `<utl_file_dir directory name>/<filename>`

To get the log in the batch immediate mode from the table, use the following script where `&reqid` is the Concurrent request ID of the Receiving Transaction Processor in the batch/immediate mode:

```
SELECT MODULE, MESSAGE_TEXT
    FROM FND_LOG_MESSAGES
    WHERE TIMESTAMP > SYSDATE - 2/24 AND
    PROCESS_ID = (
         SELECT OS_PROCESS_ID
         FROM FND_CONCURRENT_REQUESTS
         WHERE REQUEST_ID = &REQID)
    AND MODULE LIKE 'PO%'
    ORDER BY LOG_SEQUENCE ASC;
```

To place the log in online mode from the table:

1. SELECT MAX(LOG_SEQUENCE) FROM FND_LOG_MESSAGES, and record the value.
2. Perform the transaction in the online mode.
3. SELECT MAX(LOG_SEQUENCE) FROM FND_LOG_MESSAGES.
4. SELECT MODULE, MESSAGE_TEXT FROM the FND_LOG_MESSAGES WHERE LOG_SEQUENCE is between the value from step 1 and the value from step 2.

Resolving Failed Receiving Open Interface Rows

Error Messages

Oracle Purchasing may display specific error messages during interface processing. For more details on these messages, see the *Oracle Applications Messages Manual*.

Viewing Failed Transactions

For each row in the RCV_HEADERS_INTERFACE and RCV_TRANSACTIONS_INTERFACE tables that fails validation, the Receiving Open

Interface creates one or more rows with error information in the PO_INTERFACE_ERRORS table.

You can report on all rows that failed validation by using the Receiving Interface Errors report and, for ASBNs, the Purchasing Interface Errors Report. For every transaction in the interface table that fails validation, these reports list all the columns that failed validation along with the reason for the failure.

You can identify failed transactions in the interface tables by selecting rows with a PROCESS_FLAG of ERROR or PRINT. For any previously processed set of rows identified by the HEADER_INTERFACE_ID and INTERFACE_TRANSACTION_ID, only rows that failed validation remain in the interface table. As all the successfully imported rows are deleted from the RCV_TRANSACTIONS_INTERFACE table. (Successfully imported rows in the RCV_HEADERS_INTERFACE table are not deleted.)

Related Topics

Purchasing Interface Errors Report, *Oracle Purchasing User's Guide*

Receiving Interface Errors Report, *Oracle Purchasing User's Guide*

Receiving Open Interface Details for Advanced Shipment Notice Import and Receipt Transactions

This section covers the following transactions:

- ASN/ASBN Creation
- PO Receipt
- ASN Receipt
- RMA Receipt
- Internal Requisition Receipt
- Inter-Org Transfer Receipt

RCV_HEADERS_INTERFACE (RHI)

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition	Inter-Org Transfer Receipt
HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE_S.NEXTVAL	RCV_HEADERS_INTERFACE_S.NEXTVAL	RCV_HEADERS_INTERFACE_S.NEXTVAL	RCV_HEADERS_INTERFACE_S.NEXTVAL	RCV_HEADERS_INTERFACE_S.NEXTVAL	RCV_HEADERS_INTERFACE_S.NEXTVAL
GROUP_ID	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL
CREATION_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID
RECEIPT_SOURCE_CODE	VENDOR	VENDOR	VENDOR	CUSTOMER	INTERNAL ORDER	INVENTORY
ASN_TYPE	ASN / ASBN	STD	STD	STD	STD	STD

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition	Inter-Org Transfer Receipt
VALIDATION_FLAG	Y	Y	Y	Y	Y	Y
TRANSACTION_TYPE	NEW	NEW	NEW	NEW	NEW	NEW
VENDOR_NAME	Supplier name from the document (PO_VENDORS.VENDOR_NAME)	Supplier name from the document (PO_VENDORS.VENDOR_NAME)	Supplier name from the document (PO_VENDORS.VENDOR_NAME)	NULL	NULL	NULL
VENDOR_ID	PO_HEADERS_ALL.VENDOR_ID	PO_HEADERS_ALL.VENDOR_ID	PO_HEADERS_ALL.VENDOR_ID	NULL	NULL	NULL
VENDOR_SITE_CODE	Supplier site name from the document (PO_VENDOR_SITES_ALL.VENDOR_SITE_CODE)	Supplier site name from the document (PO_VENDOR_SITES_ALL.VENDOR_SITE_CODE)	Supplier site name from the document (PO_VENDOR_SITES_ALL.VENDOR_SITE_CODE)	NULL	NULL	NULL
VENDOR_SITE_ID	PO_HEADERS_ALL.VENDOR_SITE_ID	PO_HEADERS_ALL.VENDOR_SITE_ID	PO_HEADERS_ALL.VENDOR_SITE_ID	NULL	NULL	NULL

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition	Inter-Org Transfer Receipt
SHIP_TO_ORGANIZATION_CODE	Ship to organization code from the document (MTL_PARAMETERS.ORGANIZATION_CODE)	Ship to organization code from the document (MTL_PARAMETERS.ORGANIZATION_CODE)	Ship to organization code from the document (MTL_PARAMETERS.ORGANIZATION_CODE)	Ship to organization code from the document (MTL_PARAMETERS.ORGANIZATION_CODE)	Ship to organization code from the document (MTL_PARAMETERS.ORGANIZATION_CODE)	Ship to organization code from the document (MTL_PARAMETERS.ORGANIZATION_CODE)
SHIP_TO_ORGANIZATION_ID	PO_LINE_LOCATIONS_ALL.SHIP_TO_ORGANIZATION_ID	PO_LINE_LOCATIONS_ALL.SHIP_TO_ORGANIZATION_ID	PO_LINE_LOCATIONS_ALL.SHIP_TO_ORGANIZATION_ID	OE_ORDER_LINES_ALL.SHIP_FROM_ORG_ID	RCV_SHIPMENT_LINES.TO_ORGANIZATION_ID	RCV_SHIPMENT_LINES.TO_ORGANIZATION_ID
SHIPPED_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE
EXPECTED_RECEIPT_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE
CUSTOMER_PARTY_NAME	NULL	NULL	NULL	RMA CUSTOMER NAME (HZ_PARTIES.PARTY_NAME)	NULL	NULL
CUSTOMER_SITE_ID	NULL	NULL	NULL	RMA CUSTOMER SITE ID	NULL	NULL

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition	Inter-Org Transfer Receipt
				(OE_ORDER_LINES_ALL.SHIP_FROM_ORG_ID)		

RCV_TRANSACTIONS_INTERFACE (RTI)

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition Receipt	Inter-Org Transfer Receipt
INTERFACE_TRANSACTION_ID	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL
GROUP_ID	RCV_HEADERS_INTERFACE.GROUP_ID	RCV_HEADERS_INTERFACE.GROUP_ID	RCV_HEADERS_INTERFACE.GROUP_ID	RCV_HEADERS_INTERFACE.GROUP_ID	RCV_HEADERS_INTERFACE.GROUP_ID	RCV_HEADERS_INTERFACE.GROUP_ID
HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE.HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE.HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE.HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE.HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE.HEADER_INTERFACE_ID	RCV_HEADERS_INTERFACE.HEADER_INTERFACE_ID
LPN_GROUP_ID	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL	RCV_INTERFACE_GROUPS_S.NEXTVAL
CREATION_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition Receipt	Inter-Org Transfer Receipt
CREATED_BY	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID	FND_GLOBAL.LOGIN_ID
TRANSACTION_DATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE	SYSDATE
PROCESSING_MODE_CODE	BATCH	BATCH	BATCH	BATCH	BATCH	BATCH
PROCESSING_STATUS_CODE	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING
TRANSACTION_STATUS_CODE	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING
DESTINATION_TYPE_CODE	NULL	RECEIVING	RECEIVING	RECEIVING	RECEIVING	RECEIVING
VALIDATION_FLAG	Y	Y	Y	Y	Y	Y

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition Receipt	Inter-Org Transfer Receipt
TRANSACTION_TYPE	SHIP	RECEIVE	RECEIVE	RECEIVE	RECEIVE	RECEIVE
AUTO_TRANSACT_CODE	SHIP/RECEIVE/DELIVER	SHIP/RECEIVE/DELIVER	NULL	NULL	NULL	NULL
SOURCE_DOCUMENT_CODE	PO	PO	PO	RMA	REQ	INVENTORY
DOCUMENT_NUM	PO number (PO_HEADERS_ALL.SEGMENT1)	PO number (PO_HEADERS_ALL.SEGMENT1)	ASN number (RCV_SHIPMENT_HEADERS.SHIPMENT_NUM)	RMA number (OE_ORDER_HEADERS_ALL.ORDER_NUM)	REQ number (PO_REQUISITION_HEADERS_ALL.SEGMENT1)	SHIPMENT number (RCV_SHIPMENT_HEADERS.SHIPMENT_NUM)
DOCUMENT_LINE_NUM	PO line number (PO_LINES_ALL.LINE_NUN)	PO line number (PO_LINES_ALL.LINE_NUN)	NULL	RMA line number (OE_ORDER_LINES_ALL.LINE_NUM)	REQ line number (PO_REQUISITION_LINES_ALL.LINE_NUM)	NULL
DOCUMENT_SHIPMENT_LINE_NUM	NULL	PO line location number (PO_LINE_LOCATIONS_ALL.SHIPMENT_NUM)	NULL	NULL	NULL	NULL

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition Receipt	Inter-Org Transfer Receipt
ITEM_NUM	Item from the document (MTL_SYSTEM_ITEMS.SEGMENT1)	Item from the document (MTL_SYSTEM_ITEMS.SEGMENT1)	Item from the document (MTL_SYSTEM_ITEMS.SEGMENT1)	Item from the document (MTL_SYSTEM_ITEMS.SEGMENT1)	Item from the document (MTL_SYSTEM_ITEMS.SEGMENT1)	Item from the document (MTL_SYSTEM_ITEMS.SEGMENT1)
QUANTITY	Transaction quantity	Transaction quantity	Transaction quantity	Transaction quantity	Transaction quantity	Transaction quantity
UNIT_OF_MEASURE	Transaction UOM	Transaction UOM	Transaction UOM	Transaction UOM	Transaction UOM	Transaction UOM
RECEIPT_SOURCE_CODE	VENDOR	VENDOR	VENDOR	CUSTOMER	INTERNAL ORDER	INVENTORY
SUBINVENTORY	NULL	Destination subinventory (MTL_SECONDARY_INVENTORIES.SECONDARY_INVENTORY_NAME)	Destination subinventory (MTL_SECONDARY_INVENTORIES.SECONDARY_INVENTORY_NAME)	Destination subinventory (MTL_SECONDARY_INVENTORIES.SECONDARY_INVENTORY_NAME)	Destination subinventory (MTL_SECONDARY_INVENTORIES.SECONDARY_INVENTORY_NAME)	Destination subinventory (MTL_SECONDARY_INVENTORIES.SECONDARY_INVENTORY_NAME)
LOCATOR_ID	NULL	Destination locator (MTL_ITEM_LOCATIONS_KFV.CONCANTENATED_SEGMENTS)	Destination locator (MTL_ITEM_LOCATIONS_KFV.CONCANTENATED_SEGMENTS)	Destination locator (MTL_ITEM_LOCATIONS_KFV.CONCANTENATED_SEGMENTS)	Destination locator (MTL_ITEM_LOCATIONS_KFV.CONCANTENATED_SEGMENTS)	Destination locator (MTL_ITEM_LOCATIONS_KFV.CONCANTENATED_SEGMENTS)

Column Name	ASN/ASBN Creation	PO Receipt	ASN Receipt	RMA Receipt	Internal Requisition Receipt	Inter-Org Transfer Receipt
LICENSE_PLATE_NUMBER	LPN shipped by Supplier	NULL	LPN shipped by Supplier	NULL	LPN shipped from the source organization	LPN shipped from the source organization
TRANSFER_LICENSE_PLATE_NUMBER	NULL	LPN into which the material is received	LPN into which the material is received	LPN into which the material is received	LPN into which the material is received	LPN into which the material is received

The following tables are required only if item is lot controlled, or serial controlled, or the transaction involves LPNs.

MTL_TRANSACTION_LOTS_INTERFACE (MTLI)

Column Name	
TRANSACTION_INTERFACE_ID	MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL
SERIAL_TRANSACTION_TEMP_ID	MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL
PRODUCT_CODE	'RCV'
PRODUCT_TRANSACTION_ID	RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
CREATION_DATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID

Column Name	
LOT_NUMBER	Lot number being transacted
TRANSACTION_QUANTITY	Transaction quantity for current lot
PRIMARY_QUANTITY	Primary quantity for current lot

MTL_SERIAL_NUMBERS_INTERFACE (MSNI)

Column Name	
TRANSACTION_INTERFACE_ID	MTL_TRANSACTION_LOTS_INTERFACE.SERIAL_TRANSACTION_TEMP_ID OR MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL
PRODUCT_CODE	RCV
PRODUCT_TRANSACTION_ID	RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
CREATION_DATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID
FM_SERIAL_NUMBER	From serial number
TO_SERIAL_NUMBER	To serial number

WMS_LPN_INTERFACE (WLPNI)

Column Name	
SOURCE_GROUP_ID	RCV_TRANSACTIONS_INTERFACE.LPN_GROUP_ID
CREATION_DATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID
LICENSE_PLATE_NUMBER	New LPN
PARENT_LICENSE_PLATE_NUMBER	Parent LPN

The following conditions apply to ASN and receipt transactions:

- Populate the WMS_LPN_INTERFACE only if the transaction involves a new LPN or a change in nesting structure.
- Populate MTL_SERIAL_NUMBERS_INTERFACE for only serial controlled items. If an item is only serial controlled (no lot control) then MSNI.INTERFACE_TRANSACTION_ID should be MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL.
- Populate MTL_TRANSACTION_LOTS_INTERFACE for only lot-controlled items. If an item is only lot controlled (no serial control), then MTLI.SERIAL_TRANSACTION_TEMP_ID should be null.
- Populate both MTL_SERIAL_NUMBERS_INTERFACE and MTL_TRANSACTION_LOTS_INTERFACE for lot and serial controlled items.
- In all above transactions, AUTO_TRANSACT_CODE can have a value of NULL, RECEIVE, or DELIVER. If TRANSACTION_TYPE is SHIP, and AUTO_TRANSACT_CODE is RECEIVE, the system performs the receipt as part of the ASN/ASBN creation transaction. Similarly, if TRANSACTION_TYPE is SHIP or RECEIVE, and the AUTO_TRANSACT_CODE is DELIVER, then the system

performs the delivery to INVENTORY or EXPENSE destination as part of the transaction.

The system uses RTI.DESTINATION_TYPE_CODE to determine the destination type. This column should have a value of RECEIVING for standard receipts, INVENTORY for a direct receipt into inventory, and EXPENSE for a direct receipt into an expense destination.

- For all the transactions other than ASN/ASBN, if you want a set of transactions to fail when any one transaction in that set fails, then populate the same LPN_GROUP_ID for all the rows in that set. LPN_GROUP_ID should not have same value across HEADER_INTERFACE_IDs. For ASN/ASBN transactions, the profile option RCV: Fail All ASN Lines if One Line Fails controls this situation.

- If you transaction only one serial number then FM_SERIAL_NUMBER and TO_SERIAL_NUMBER in table MTL_SERIAL_NUMBERS_INTERFACE must have the same value.

Receiving Open Interface Data Details for Post Receipt Transactions

This section covers information about the following transactions:

- Transfer
- Accept/Reject
- Deliver
- Correct
- Return to Vendor
- Return to Receiving
- Return to Customer

You can source these transactions from any source document. These transactions do not require a record in RCV_HEADERS_INTERFACE_TABLE. Do not insert records in RCV_HEADERS_INTERFACE for these transactions.

RCV_TRANSACTIONS_INTERFACE (RTI)

Column Name	Post RECEIPT Transactions (TRANSFER ACCEPT/REJECT DELIVER CORRECT RETURN TO VENDOR RETURN TO CUSTOMER RETURN TO RECEIVING)
INTERFACE_TRANSACTION_ID	RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL
GROUP_ID	RCV_INTERFACE_GROUPS_S.NEXTVAL
HEADER_INTERFACE_ID	NULL
LPN_GROUP_ID	RCV_INTERFACE_GROUPS_S.NEXTVAL
CREATION_DATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID
TRANSACTION_DATE	SYSDATE
PROCESSING_MODE_CODE	BATCH
PROCESSING_STATUS_CODE	PENDING
TRANSACTION_STATUS_CODE	PENDING
DESTINATION_TYPE_CODE	RECEIVING OR INVENTORY OR EXPENSE
TRANSACTION_TYPE	TRANSFER / ACCEPT / REJECT / DELIVER / CORRECT / RETURN TO VENDOR / RETURN TO CUSTOMER / RETURN TO RECEIVING
VALIDATION_FLAG	Y

Column Name	Post RECEIPT Transactions (TRANSFER ACCEPT/REJECT DELIVER CORRECT RETURN TO VENDOR RETURN TO CUSTOMER RETURN TO RECEIVING)
QUANTITY	Transaction quantity. For negative corrections give a negative value.
UNIT_OF_MEASURE	Transaction UOM.
RECEIPT_SOURCE_CODE	VENDOR / CUSTOMER / INTERNAL ORDER / INVENTORY
SOURCE_DOCUMENT_CODE	PO / RMA / REQ / INVENTORY
SUBINVENTORY	Destination subinventory (MTL_SECONDARY_INVENTORIES.SECONDARY_INVENTORY_NAME)
LOCATOR_ID	Destination locator (MTL_ITEM_LOCATIONS_KFV.CONCANTENATED_SEGMENTS)
LICENSE_PLATE_NUMBER	Source LPN (Destination LPN for correction)
TRANSFER_LICENSE_PLATE_NUMBER	Destination LPN (Source LPN for correction)
PARENT_TRANSACTION_ID	Transaction ID for the parent transaction (RCV_TRANSACTIONS.TRANSACTION_ID)

The following tables are required only if item the is lot controlled, if the item is serial controlled, or if the transaction involves LPNs.

MTL_TRANSACTION_LOTS_INTERFACE (MTLI)

Column Name	Record1
TRANSACTION_INTERFACE_ID	MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL
SERIAL_TRANSACTION_TEMP_ID	MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL

Column Name	Record1
PRODUCT_CODE	RCV
PRODUCT_TRANSACTION_ID	RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
CREATION_DATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID
LOT_NUMBER	Lot number being transacted
TRANSACTION_QUANTITY	Transaction quantity for current lot
PRIMARY_QUANTITY	Primary quantity for current lot

MTL_SERIAL_NUMBERS_INTERFACE (MSNI)

Column Name	Record1
TRANSACTION_INTERFACE_ID	MTL_TRANSACTION_LOTS_INTERFACE.SERIAL_TRANSACTION_TEMP_ID OR MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL
PRODUCT_CODE	RCV
PRODUCT_TRANSACTION_ID	RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
CREATION_DATE	SYSDATE

Column Name	Record1
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID
FM_SERIAL_NUMBER	From serial number
TO_SERIAL_NUMBER	To Serial Number

WMS_LPN_INTERFACE (WLPNI)

Column Name	Record1
SOURCE_GROUP_ID	RCV_TRANSACTIONS_INTERFACE.LPN_GROUP_ID
CREATION_DATE	SYSDATE
CREATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_DATE	SYSDATE
LAST_UPDATED_BY	FND_GLOBAL.USER_ID
LAST_UPDATE_LOGIN	FND_GLOBAL.LOGIN_ID
LICENSE_PLATE_NUMBER	New LPN
PARENT_LICENSE_PLATE_NUMBER	Parent LPN

The following conditions apply to post-receipt transactions:

- Populate WMS_LPN_INTERFACE only if the transaction involves a new LPN or a change in the nesting structure.

- Populate MTL_SERIAL_NUMBERS_INTERFACE only for serial controlled items. If an item is only serial controlled (No lot control) then MSNI.INTERFACE_TRANSACTION_ID should be MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL.

- Populate MTL_TRANSACTION_LOTS_INTERFACE only for lot-controlled items. If an item is only lot controlled (No serial control) then MTLI.SERIAL_TRANSACTION_TEMP_ID should be null.

- Populate both MTL_SERIAL_NUMBERS_INTERFACE and MTL_TRANSACTION_LOTS_INTERFACE for lot and serial controlled items.

- For all the transactions other than ASN/ASBN transactions, if you want a set of transactions to fail when any one transaction fails, then populate the same LPN_GROUP_ID for all the rows in that set. The LPN_GROUP_ID should not have the same value across multiple GROUP_IDs.

- If only one Serial is being transacted, FM_SERIAL_NUMBER and TO_SERIAL_NUMBER in table MTL_SERIAL_NUMBERS_INTERFACE must have the same value.

Sample Scripts

This section provides information documents that you can receive using the Receiving Open Interface.

Purchase Order Sample Script

This script provides a sample SELECT statement to retrieve document information for a purchase order.

- To get header information:

```
SELECT * FROM PO_HEADERS_ALL POH WHERE SEGMENT1 = 'DOCUMENT_NUMBER';
```

- To get line information:

```
SELECT * FROM PO_LINES_ALL WHERE PO_HEADER_ID = PO_HEADER_ID FROM
THE FIRST SQL;
```

- To get distribution (delivery) information:

```
SELECT * FROM PO_DISTRIBUTIONS_ALL WHERE PO_HEADER_ID = PO_HEADER_ID
FROM THE FIRST SQL;
```

The SQL to get PO shipments that are eligible for Receiving is as follows:

Note: The SQL uses X_INCLUDE_CLOSED_PO, which is the value of

the profile option RCV: Default Include Closed PO Option. It also uses X_PO_NUM which is the PO number (PO_HEADERS_ALL.SEGMENT1).

```
SELECT PLL.LINE_LOCATION_ID,
  PLL.UNIT_MEAS_LOOKUP_CODE,
  PLL.UNIT_OF_MEASURE_CLASS,
  NVL(PLL.PROMISED_DATE,   PLL.NEED_BY_DATE) PROMISED_DATE,
  PLL.SHIP_TO_ORGANIZATION_ID,
  PLL.QUANTITY QUANTITY_ORDERED,
  PLL.QUANTITY_SHIPPED,
  PLL.RECEIPT_DAYS_EXCEPTION_CODE,
  PLL.QTY_RCV_TOLERANCE,
  PLL.QTY_RCV_EXCEPTION_CODE,
  PLL.DAYS_EARLY_RECEIPT_ALLOWED,
  PLL.DAYS_LATE_RECEIPT_ALLOWED,
  NVL(PLL.PRICE_OVERRIDE,   PL.UNIT_PRICE) UNIT_PRICE,
  PLL.MATCH_OPTION,
  PL.CATEGORY_ID,
  NVL(PLL.DESCRIPTION,   PL.ITEM_DESCRIPTION) ITEM_DESCRIPTION,
  PL.PO_LINE_ID,
  PH.CURRENCY_CODE,
  PH.RATE_TYPE,
  POD.PO_DISTRIBUTION_ID,
  POD.CODE_COMBINATION_ID,
  POD.REQ_DISTRIBUTION_ID,
  POD.DELIVER_TO_LOCATION_ID,
  POD.DELIVER_TO_PERSON_ID,
  POD.RATE_DATE,
  POD.RATE,
  POD.DESTINATION_TYPE_CODE,
  POD.DESTINATION_ORGANIZATION_ID,
  POD.DESTINATION_SUBINVENTORY,
  POD.WIP_ENTITY_ID,
  POD.WIP_OPERATION_SEQ_NUM,
  POD.WIP_RESOURCE_SEQ_NUM,
  POD.WIP_REPETITIVE_SCHEDULE_ID,
  POD.WIP_LINE_ID,
  POD.BOM_RESOURCE_ID,
  POD.USSGL_TRANSACTION_CODE,
  PLL.SHIP_TO_LOCATION_ID,
  NVL(PLL.ENFORCE_SHIP_TO_LOCATION_CODE,   'NONE')
ENFORCE_SHIP_TO_LOCATION_CODE,
  TO_NUMBER(NULL) SHIPMENT_LINE_ID,
  PL.ITEM_ID
FROM PO_DISTRIBUTIONS_ALL POD,
  PO_LINE_LOCATIONS_ALL PLL,
  PO_LINES_ALL PL,
  PO_HEADERS_ALL PH
WHERE PH.SEGMENT1 = X_PO_NUM
 AND PL.PO_HEADER_ID = PH.PO_HEADER_ID
 AND PLL.PO_LINE_ID = PL.PO_LINE_ID
 AND POD.LINE_LOCATION_ID = PLL.LINE_LOCATION_ID
 AND NVL(PLL.APPROVED_FLAG,   'N') = 'Y'
 AND NVL(PLL.CANCEL_FLAG,   'N') = 'N'
 AND((NVL(X_INCLUDE_CLOSED_PO,   'N') = 'Y'
 AND NVL(PLL.CLOSED_CODE,   'OPEN') <> 'FINALLY CLOSED')
OR(NVL(X_INCLUDE_CLOSED_PO,   'N') = 'N'
 AND(NVL(PLL.CLOSED_CODE,   'OPEN') NOT IN('FINALLY CLOSED',   'CLOSED',
'CLOSED FOR RECEIVING'))))
 AND PLL.SHIPMENT_TYPE IN('STANDARD',   'BLANKET',   'SCHEDULED',
'PREPAYMENT')
ORDER BY NVL(PLL.PROMISED_DATE,   PLL.NEED_BY_DATE);
```

Sample Script to get ASN Information

This section provides a sample select statement to retrieve document information for an ASN.

The SQL to get ASN shipments is as follows:

> **Note:** The SQL uses X_ASN_NUMBER. This is the ASN shipment number.

```
SELECT RSL.PO_LINE_LOCATION_ID,
  PLL.UNIT_MEAS_LOOKUP_CODE,
  PLL.UNIT_OF_MEASURE_CLASS,
  NVL(PLL.PROMISED_DATE,   PLL.NEED_BY_DATE)
PROMISED_DATE,
  RSL.TO_ORGANIZATION_ID SHIP_TO_ORGANIZATION_ID,
  PLL.QUANTITY QUANTITY_ORDERED,
  RSL.QUANTITY_SHIPPED,
  PLL.RECEIPT_DAYS_EXCEPTION_CODE,
  PLL.QTY_RCV_TOLERANCE,
  PLL.QTY_RCV_EXCEPTION_CODE,
  PLL.DAYS_EARLY_RECEIPT_ALLOWED,
  PLL.DAYS_LATE_RECEIPT_ALLOWED,
  NVL(PLL.PRICE_OVERRIDE,   PL.UNIT_PRICE) UNIT_PRICE,
  PLL.MATCH_OPTION,
  RSL.CATEGORY_ID,
  RSL.ITEM_DESCRIPTION,
  PL.PO_LINE_ID,
  PH.CURRENCY_CODE,
  PH.RATE_TYPE,
  POD.PO_DISTRIBUTION_ID,
  POD.CODE_COMBINATION_ID,
  POD.REQ_DISTRIBUTION_ID,
  POD.DELIVER_TO_LOCATION_ID,
  POD.DELIVER_TO_PERSON_ID,
  POD.RATE_DATE,
  POD.RATE,
  POD.DESTINATION_TYPE_CODE,
  POD.DESTINATION_ORGANIZATION_ID,
  POD.DESTINATION_SUBINVENTORY,
  POD.WIP_ENTITY_ID,
  POD.WIP_OPERATION_SEQ_NUM,
  POD.WIP_RESOURCE_SEQ_NUM,
  POD.WIP_REPETITIVE_SCHEDULE_ID,
  POD.WIP_LINE_ID,
  POD.BOM_RESOURCE_ID,
  POD.USSGL_TRANSACTION_CODE,
  PLL.SHIP_TO_LOCATION_ID,
  NVL(PLL.ENFORCE_SHIP_TO_LOCATION_CODE,   'NONE')
ENFORCE_SHIP_TO_LOCATION_CODE,
  RSL.SHIPMENT_LINE_ID,
  RSL.ITEM_ID
FROM PO_DISTRIBUTIONS POD,
  PO_LINE_LOCATIONS PLL,
  PO_LINES PL,
  PO_HEADERS PH,
  RCV_SHIPMENT_LINES RSL,
  RCV_SHIPMENT_HEADERS RSH
WHERE RSH.SHIPMENT_NUM = X_ASN_NUMBER
 AND RSL.SHIPMENT_HEADER_ID = RSH.SHIPMENT_HEADER_ID
 AND RSL.SHIPMENT_LINE_ID = NVL(V_SHIPMENT_LINE_ID,
RSL.SHIPMENT_LINE_ID)
 AND RSL.PO_HEADER_ID = PH.PO_HEADER_ID
 AND PH.PO_HEADER_ID = NVL(V_PO_HEADER_ID,
PH.PO_HEADER_ID)
 AND((V_PO_HEADER_ID IS NOT NULL
 AND PL.LINE_NUM = NVL(V_PO_LINE_NUM,   PL.LINE_NUM)
 AND PLL.SHIPMENT_NUM = NVL(V_PO_SHIPMENT_NUM,
PLL.SHIPMENT_NUM)
```

```
AND POD.DISTRIBUTION_NUM = NVL(V_PO_DISTRIBUTION_NUM,
POD.DISTRIBUTION_NUM)) OR V_PO_HEADER_ID IS NULL)
 AND POD.PO_HEADER_ID = PH.PO_HEADER_ID
 AND POD.LINE_LOCATION_ID = PLL.LINE_LOCATION_ID
 AND RSL.PO_LINE_ID = PL.PO_LINE_ID
 AND RSL.PO_LINE_LOCATION_ID = PLL.LINE_LOCATION_ID
 AND NVL(RSL.PO_RELEASE_ID,   0) = NVL(V_PO_RELEASE_ID,
NVL(RSL.PO_RELEASE_ID,   0))
 AND NVL(RSL.ITEM_ID,   0) = NVL(V_ITEM_ID,
NVL(RSL.ITEM_ID,   0))
 AND RSL.TO_ORGANIZATION_ID = NVL(V_SHIP_TO_ORG_ID,
RSL.TO_ORGANIZATION_ID)
 AND(NVL(RSL.SHIPMENT_LINE_STATUS_CODE,   'EXPECTED') <>
'FULLY RECEIVED')
 AND RSH.RECEIPT_SOURCE_CODE = 'VENDOR'
 AND NVL(RSL.ASN_LINE_FLAG,   'N') = 'Y'
 AND RSL.SHIP_TO_LOCATION_ID = NVL(V_SHIP_TO_LOCATION_ID,
RSL.SHIP_TO_LOCATION_ID)
 AND PLL.PO_LINE_ID = PL.PO_LINE_ID
 AND NVL(PLL.PO_RELEASE_ID,   0) = NVL(RSL.PO_RELEASE_ID,
NVL(PLL.PO_RELEASE_ID,   0))
 AND NVL(PLL.APPROVED_FLAG,   'N') = 'Y'
 AND NVL(PLL.CANCEL_FLAG,   'N') = 'N'
 AND(((NVL(V_INCLUDE_CLOSED_PO,   'N') = 'Y')
 AND(NVL(PLL.CLOSED_CODE,   'OPEN') <> 'FINALLY CLOSED'))
OR((NVL(V_INCLUDE_CLOSED_PO,   'N') = 'N')
 AND(NVL(PLL.CLOSED_CODE,   'OPEN') NOT IN('FINALLY
CLOSED',   'CLOSED',   'CLOSED FOR RECEIVING'))))
 AND PLL.SHIPMENT_TYPE IN('STANDARD',   'BLANKET',
'SCHEDULED')
 AND PLL.SHIP_TO_ORGANIZATION_ID =
NVL(RSL.TO_ORGANIZATION_ID,   PLL.SHIP_TO_ORGANIZATION_ID)
 AND PLL.SHIP_TO_LOCATION_ID =
NVL(RSL.SHIP_TO_LOCATION_ID,   PLL.SHIP_TO_LOCATION_ID)
 AND NVL(PL.VENDOR_PRODUCT_NUM,   '-999') =
NVL(V_VENDOR_PRODUCT_NUM,   NVL(PL.VENDOR_PRODUCT_NUM,
'-999'));
```

Sample Script to Get Internal Requisition Information

This section provides a sample select statement to retrieve document information for an internal requisition.

```
SELECT REQUISITION_LINE_ID FROM PO_REQUISITION_LINES_ALL WHERE
REQUISITION_HEADER_ID = (SELECT REQUISITION_HEADER_ID FROM
PO_REQUISITION_HEADERS_ALL WHERE SEGMENT1 = 'REQ NUMBER';
```

```
X_REQ_LINE_ID = THE REQUISITION_LINE_ID OF THE REQUISITION LINE
```

```
SELECT RSH.SHIPMENT_HEADER_ID SHIPMENT_HEADER_ID,
  RSH.SHIPMENT_NUM SHIPMENT_NUM,
  RSL.SHIPMENT_LINE_ID SHIPMENT_LINE_ID,
  RSL.ITEM_ID ITEM_ID,
  RSL.ITEM_DESCRIPTION ITEM_DESCRIPTION,
  RSL.TO_ORGANIZATION_ID TO_ORGANIZATION_ID,
  RSL.FROM_ORGANIZATION_ID FROM_ORGANIZATION_ID,
  RSL.ROUTING_HEADER_ID ROUTING_HEADER_ID,
  RSL.CATEGORY_ID CATEGORY_ID,
  RSH.CURRENCY_CODE CURRENCY_CODE,
  RSH.CONVERSION_RATE CURRENCY_CONVERSION_RATE,
  RSH.CONVERSION_RATE_TYPE CURRENCY_CONVERSION_TYPE,
  RSH.CONVERSION_DATE CURRENCY_CONVERSION_DATE,
  RSL.TO_SUBINVENTORY TO_SUBINVENTORY,
  RSL.SHIP_TO_LOCATION_ID SHIP_TO_LOCATION_ID,
  RSL.DELIVER_TO_LOCATION_ID DELIVER_TO_LOCATION_ID,
  RSL.DELIVER_TO_PERSON_ID DELIVER_TO_PERSON_ID,
  RSL.USSGL_TRANSACTION_CODE USSGL_TRANSACTION_CODE,
  RSL.DESTINATION_TYPE_CODE DESTINATION_TYPE_CODE,
  RSL.DESTINATION_CONTEXT DESTINATION_CONTEXT,
  RSL.UNIT_OF_MEASURE UNIT_OF_MEASURE,
  RSL.PRIMARY_UNIT_OF_MEASURE PRIMARY_UNIT_OF_MEASURE,
  RSL.REQUISITION_LINE_ID REQUISITION_LINE_ID,
  RSL.PO_LINE_LOCATION_ID PO_LINE_LOCATION_ID,
  RSL.EMPLOYEE_ID EMPLOYEE_ID
FROM RCV_SHIPMENT_HEADERS RSH,
  RCV_SHIPMENT_LINES RSL,
  PO_REQUISITION_LINES_ALL PORL
WHERE PORL.REQUISITION_LINE_ID = X_REQ_LINE_ID
 AND RSL.SHIPMENT_HEADER_ID = RSH.SHIPMENT_HEADER_ID
 AND(NVL(RSL.SHIPMENT_LINE_STATUS_CODE,   'EXPECTED') <> 'FULLY
RECEIVED')
 AND RSH.RECEIPT_SOURCE_CODE = 'INTERNAL ORDER';
```

Sample Script to get Inter-Organization Transfer Shipment Information

This section provides a sample select statement to retrieve information about intransit shipments that are eligible for receiving.

Note: X_SHIPMENT_NUM = Interorganization shipment Number.

```
SELECT RSH.SHIPMENT_HEADER_ID SHIPMENT_HEADER_ID,
  RSH.SHIPMENT_NUM SHIPMENT_NUM,
  RSL.SHIPMENT_LINE_ID SHIPMENT_LINE_ID,
  RSL.ITEM_ID ITEM_ID,
  RSL.ITEM_DESCRIPTION ITEM_DESCRIPTION,
  RSL.TO_ORGANIZATION_ID TO_ORGANIZATION_ID,
  RSL.FROM_ORGANIZATION_ID FROM_ORGANIZATION_ID,
  RSL.ROUTING_HEADER_ID ROUTING_HEADER_ID,
  RSL.CATEGORY_ID CATEGORY_ID,
  RSH.CURRENCY_CODE CURRENCY_CODE,
  RSH.CONVERSION_RATE CURRENCY_CONVERSION_RATE,
  RSH.CONVERSION_RATE_TYPE CURRENCY_CONVERSION_TYPE,
  RSH.CONVERSION_DATE CURRENCY_CONVERSION_DATE,
  RSL.TO_SUBINVENTORY TO_SUBINVENTORY,
  RSL.SHIP_TO_LOCATION_ID SHIP_TO_LOCATION_ID,
  RSL.DELIVER_TO_LOCATION_ID DELIVER_TO_LOCATION_ID,
  RSL.DELIVER_TO_PERSON_ID DELIVER_TO_PERSON_ID,
  RSL.USSGL_TRANSACTION_CODE USSGL_TRANSACTION_CODE,
  RSL.DESTINATION_TYPE_CODE DESTINATION_TYPE_CODE,
  RSL.DESTINATION_CONTEXT DESTINATION_CONTEXT,
  RSL.UNIT_OF_MEASURE UNIT_OF_MEASURE,
  RSL.PRIMARY_UNIT_OF_MEASURE PRIMARY_UNIT_OF_MEASURE
FROM RCV_SHIPMENT_HEADERS RSH,
  RCV_SHIPMENT_LINES RSL
WHERE RSH.SHIPMENT_NUM = X_SHIPMENT_NUM
 AND RSL.SHIPMENT_HEADER_ID = RSH.SHIPMENT_HEADER_ID
 AND(NVL(RSL.SHIPMENT_LINE_STATUS_CODE,   'EXPECTED') <>
'FULLY RECEIVED')
 AND RSH.RECEIPT_SOURCE_CODE = 'INVENTORY';
```

Sample Script to Get Return Material Authorization (RMA) Information

This section provides a sample select statement to retrieve information about RMA lines that are eligible for receiving.

Note: X_RMA_NUMBER = RMA Number

```
SELECT NVL(OEL.SHIP_TO_ORG_ID,   OEH.SHIP_TO_ORG_ID)
CUSTOMER_SITE_ID,
  NVL(OEL.SHIP_FROM_ORG_ID,   OEH.SHIP_FROM_ORG_ID)
TO_ORGANIZATION_ID,
  NVL(OEL.SOLD_TO_ORG_ID,   OEH.SOLD_TO_ORG_ID)
CUSTOMER_ID,
  NVL(OEL.PROMISE_DATE,   OEL.REQUEST_DATE)
EXPECTED_RECEIPT_DATE,
  OEL.ORDERED_QUANTITY ORDERED_QTY,
  'N' ENFORCE_SHIP_TO_LOCATION_CODE,
  OEL.DELIVER_TO_CONTACT_ID DELIVER_TO_PERSON_ID,
  OEL.DELIVER_TO_ORG_ID DELIVER_TO_LOCATION_ID,
  OEL.HEADER_ID OE_ORDER_HEADER_ID,
  OEL.LINE_ID OE_ORDER_LINE_ID,
  OEH.ORDER_NUMBER OE_ORDER_NUM,
  OEL.LINE_NUMBER OE_ORDER_LINE_NUM,
  OEL.INVENTORY_ITEM_ID ITEM_ID,
  MUM.UNIT_OF_MEASURE,
  MSI.DESCRIPTION DESCRIPTION
FROM OE_ORDER_HEADERS_ALL OEH,
  OE_ORDER_LINES_ALL OEL,
  OE_TRANSACTION_TYPES_ALL OLT,
  OE_TRANSACTION_TYPES_TL T,
  MTL_UNITS_OF_MEASURE_TL MUM,
  MTL_SYSTEM_ITEMS MSI
WHERE OEH.ORDER_NUMBER = X_RMA_NUMBER
 AND OEH.HEADER_ID = OEL.HEADER_ID
 AND OEH.OPEN_FLAG = 'Y'
 AND OEL.LINE_CATEGORY_CODE = 'RETURN'
 AND OEL.OPEN_FLAG = 'Y'
 AND OEL.LINE_TYPE_ID = OLT.TRANSACTION_TYPE_ID
 AND OLT.TRANSACTION_TYPE_CODE = 'LINE'
 AND OLT.TRANSACTION_TYPE_ID = T.TRANSACTION_TYPE_ID
 AND T.LANGUAGE = USERENV('LANG')
 AND MSI.ORGANIZATION_ID =
OE_SYS_PARAMETERS.VALUE('MASTER_ORGANIZATION_ID',
OEL.ORG_ID)
 AND MSI.INVENTORY_ITEM_ID = OEL.INVENTORY_ITEM_ID
 AND OEL.BOOKED_FLAG = 'Y'
 AND OEL.ORDERED_QUANTITY > NVL(OEL.SHIPPED_QUANTITY,   0)
 AND OEL.FLOW_STATUS_CODE = 'AWAITING_RETURN'
 AND OEL.ORDER_QUANTITY_UOM = MUM.UOM_CODE
 AND MUM.LANGUAGE = USERENV('LANG')
ORDER BY EXPECTED_RECEIPT_DATE;
```

Sample Script to Populate the Receiving Open Interface for a Receiving Transaction for a Lot and Serial Controlled Item with an LPN.

The following sample script is for a receipt of a PO (document number 5246). The item CM13139 is a lot and serial controlled item, and the quantity to be received is 20. The ship to organization code is V1, and the ship-to location code is V1-New York City. The supplier is Consolidated Supplies, and the supplier site is Dallas-ERS. The material is received into LPN1. Serials SER1-SER15 belong to lot number LOT1, and serials SER16 and SER20 belong to lot number LOT2.

```
DECLARE
   L_RTI_ID NUMBER;
   L_LPN_ID NUMBER;

BEGIN
/* INSERT INTO RCV_HEADERS_INTERFACE */

INSERT INTO RCV_HEADERS_INTERFACE
(HEADER_INTERFACE_ID,
 GROUP_ID,
 PROCESSING_STATUS_CODE,
 RECEIPT_SOURCE_CODE,
 ASN_TYPE,
 TRANSACTION_TYPE,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 LAST_UPDATE_LOGIN,
 CREATION_DATE,
 CREATED_BY,
 SHIPPED_DATE,
 SHIPMENT_NUM,
 VENDOR_NAME,
 VENDOR_SITE_CODE,
 VALIDATION_FLAG,
 SHIP_TO_ORGANIZATION_CODE,
 EXPECTED_RECEIPT_DATE
)
SELECT
 PO.RCV_HEADERS_INTERFACE_S.NEXTVAL,
 PO.RCV_INTERFACE_GROUPS_S.NEXTVAL,
 'PENDING',
 'VENDOR',
 'STD',
 'NEW',
 SYSDATE,
 1,
 1,
 SYSDATE,
 1,
 SYSDATE,
 'CONSOLIDATED SUPPLIES',
 'DALLAS-ERS',
 'Y',
 'V1',
 SYSDATE+5
FROM DUAL;

/* INSERT INTO RCV_TRANSACTIONS_INTERFACE */

INSERT INTO RCV_TRANSACTIONS_INTERFACE
(INTERFACE_TRANSACTION_ID,
 HEADER_INTERFACE_ID,
 GROUP_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 TRANSACTION_TYPE,
 TRANSACTION_DATE,
 PROCESSING_STATUS_CODE,
```

```
PROCESSING_MODE_CODE,
 TRANSACTION_STATUS_CODE,
 QUANTITY,
 UNIT_OF_MEASURE,
 AUTO_TRANSACT_CODE,
 RECEIPT_SOURCE_CODE,
 SOURCE_DOCUMENT_CODE,
 DOCUMENT_NUM,
 SHIP_TO_LOCATION_CODE,
 VENDOR_NAME,
 VALIDATION_FLAG,
 TO_ORGANIZATION_CODE,
 ITEM_NUM,
 LPN_GROUP_ID,
 TRANSFER_LICENSE_PLATE_NUMBER
)
SELECT
 PO.RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL,
 PO.RCV_HEADERS_INTERFACE_S.CURRVAL,
 PO.RCV_INTERFACE_GROUPS_S.CURRVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'RECEIVE',
 SYSDATE,
 'PENDING',
 'BATCH',
 'PENDING',
 20,
 'EACH',
 'RECEIVE',
 'VENDOR',
 'PO',
 '5246',
 'V1- NEW YORK CITY',
 'CONSOLIDATED SUPPLIES',
 'Y',
 'V1',
 'CM13139',
 PO.RCV_INTERFACE_GROUPS_S.NEXTVAL,
 'LPN1'
FROM DUAL;

/* INSERT FIRST LOT INTO MTL_TRANSACTION_LOTS_INTERFACE */



INSERT INTO MTL_TRANSACTION_LOTS_INTERFACE
(TRANSACTION_INTERFACE_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 LOT_NUMBER,
 TRANSACTION_QUANTITY,
 PRIMARY_QUANTITY,
 SERIAL_TRANSACTION_TEMP_ID,
```

```
PRODUCT_CODE,
 PRODUCT_TRANSACTION_ID
)
SELECT
 MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'LOT1', --First Lot Number
 15,
 15,
 MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
 'RCV',
 PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;


/* INSERT SERIALS CORRESPONDING TO FIRST LOT INTO
MTL_SERIAL_NUMBERS_INTERFACE */


INSERT INTO MTL_SERIAL_NUMBERS_INTERFACE
(TRANSACTION_INTERFACE_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 FM_SERIAL_NUMBER,
 TO_SERIAL_NUMBER,
 PRODUCT_CODE,
 PRODUCT_TRANSACTION_ID
)
SELECT
 MTL_MATERIAL_TRANSACTIONS_S.CURRVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'SER1', --From Serial Number
 'SER15', --To Serial Number
 'RCV',
 PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;

/* INSERT SECOND LOT INTO MTL_TRANSACTION_LOTS_INTERFACE */

INSERT INTO MTL_TRANSACTION_LOTS_INTERFACE
(TRANSACTION_INTERFACE_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 LOT_NUMBER,
 TRANSACTION_QUANTITY,
```

```
PRIMARY_QUANTITY,
 SERIAL_TRANSACTION_TEMP_ID,
 PRODUCT_CODE,
 PRODUCT_TRANSACTION_ID
)
SELECT
 MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'LOT2', --Second Lot Number
 5,
 5,
 MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
 'RCV',
 PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;


/* INSERT SERIALS CORRESPONDING TO SECOND LOT INTO
MTL_SERIAL_NUMBERS_INTERFACE */

INSERT INTO MTL_SERIAL_NUMBERS_INTERFACE
(TRANSACTION_INTERFACE_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 FM_SERIAL_NUMBER,
 TO_SERIAL_NUMBER,
 PRODUCT_CODE,
 PRODUCT_TRANSACTION_ID
)
SELECT
 MTL_MATERIAL_TRANSACTIONS_S.CURRVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'SER16', --From Serial Number
 'SER20', --To Serial Number
 'RCV',
 PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;

/* INSERT INTO WMS_LPN_INTERFACE FOR A NEW LICENSE PLATE NUMBER. */

INSERT INTO WMS_LPN_INTERFACE
(SOURCE_GROUP_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 LAST_UPDATE_LOGIN,
 CREATION_DATE,
 CREATED_BY,
 LICENSE_PLATE_NUMBER)
```

```
VALUES
(PO.RCV_INTERFACE_GROUPS_S.CURRVAL, --This should join to
RCV_TRANSACTIONS_INTERFACE.LPN_GROUP_ID
 SYSDATE,
 1,
 1,
 SYSDATE,
 1,
 'LPN1'
);

COMMIT;

EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR');
END;
/
```

Sample Script to Populate the Receiving Open Interface for a Correct transaction for a Lot and Serial Controlled item with LPN.

The following script is for a positive correction for the Receive transaction that you performed in the previous section. Quantity 2 is added to the lot number LOT2.

> **Note:** For these transactions, you do not need to populate the RCV_HEADERS_INTERFACE table.

```
DECLARE
  L_PARENT_ID NUMBER;
  L_LPN_ID NUMBER;

BEGIN
/* INSERT INTO RCV_TRANSACTIONS_INTERFACE. */

/* SELECT TRANSACTION_ID FROM RCV_TRANSACTIONS FOR THE RECEIVE
TRANSACTION INTO L_PARENT_ID AND POPULATE PARENT_TRANSACTION_ID IN THE
SQL BELOW. ALSO SELECT TRANSFER_LPN_ID OF THE RECEIPT TRANSACTION INTO
L_LPN_ID */

INSERT INTO RCV_TRANSACTIONS_INTERFACE
(INTERFACE_TRANSACTION_ID,
 GROUP_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 TRANSACTION_TYPE,
 TRANSACTION_DATE,
 PROCESSING_STATUS_CODE,
 PROCESSING_MODE_CODE,
 TRANSACTION_STATUS_CODE,
 QUANTITY,
 RECEIPT_SOURCE_CODE,
 SOURCE_DOCUMENT_CODE,
 DOCUMENT_NUM,
 VALIDATION_FLAG,
 TO_ORGANIZATION_ID,
 PARENT_TRANSACTION_ID,
 TRANSFER_LPN_ID
)
SELECT
 PO.RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL,
 PO.RCV_INTERFACE_GROUPS_S.NEXTVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'CORRECT',
 SYSDATE,
 'PENDING',
 'BATCH',
 'PENDING',
 2,
 'VENDOR',
 'PO',
 '5246',
 'Y',
 204,
 L_PARENT_ID,
 L_LPN_ID
FROM DUAL;

/* INSERT INTO MTL_TRANSACTION_LOTS_INTERFACE */

 INSERT INTO MTL_TRANSACTION_LOTS_INTERFACE
 (TRANSACTION_INTERFACE_ID,
```

```
LAST_UPDATE_DATE,
  LAST_UPDATED_BY,
  CREATION_DATE,
  CREATED_BY,
  LAST_UPDATE_LOGIN,
  LOT_NUMBER,
  TRANSACTION_QUANTITY,
  PRIMARY_QUANTITY,
  SERIAL_TRANSACTION_TEMP_ID,
  PRODUCT_CODE,
  PRODUCT_TRANSACTION_ID
 )
 SELECT
  MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
  SYSDATE,
  1,
  SYSDATE,
  1,
  1,
  'LOT2',
  2,
  2,
  MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
  'RCV',
  PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;


/* INSERT INTO MTL_SERIAL_NUMBERS_INTERFACE */

INSERT INTO MTL_SERIAL_NUMBERS_INTERFACE
(TRANSACTION_INTERFACE_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 FM_SERIAL_NUMBER,
 TO_SERIAL_NUMBER,
 PRODUCT_CODE,
 PRODUCT_TRANSACTION_ID
)
SELECT
 MTL_MATERIAL_TRANSACTIONS_S.CURRVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'SER21', --From Serial Number
 'SER22', --To Serial Number
 'RCV',
 PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;

COMMIT;

EXCEPTION
  WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('ERROR');
END;
/
```

The following script is for a negative correction for the Receive transaction that is done in the section 5.1.6. Quantity 2 is removed from the lot number LOT1.

```
DECLARE
  L_PARENT_ID NUMBER;
  L_LPN_ID NUMBER;

BEGIN
/* INSERT INTO RCV_TRANSACTIONS_INTERFACE. */

/* SELECT TRANSACTION_ID FROM RCV_TRANSACTIONS FOR THE RECEIVE
TRANSACTION INTO L_PARENT_ID AND POPULATE PARENT_TRANSACTION_ID IN THE
SQL BELOW. ALSO SELECT TRANSFER_LPN_ID OF THE RECEIPT TRANSACTION INTO
L_LPN_ID */

INSERT INTO RCV_TRANSACTIONS_INTERFACE
(INTERFACE_TRANSACTION_ID,
 GROUP_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 TRANSACTION_TYPE,
 TRANSACTION_DATE,
 PROCESSING_STATUS_CODE,
 PROCESSING_MODE_CODE,
 TRANSACTION_STATUS_CODE,
 QUANTITY,
 RECEIPT_SOURCE_CODE,
 SOURCE_DOCUMENT_CODE,
 DOCUMENT_NUM,
 VALIDATION_FLAG,
 TO_ORGANIZATION_ID,
 PARENT_TRANSACTION_ID,
 TRANSFER_LPN_ID
)
SELECT
 PO.RCV_TRANSACTIONS_INTERFACE_S.NEXTVAL,
 PO.RCV_INTERFACE_GROUPS_S.NEXTVAL,
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'CORRECT',
 SYSDATE,
 'PENDING',
 'BATCH',
 'PENDING',
 2,
 'VENDOR',
 'PO',
 '5246',
 'Y',
 204,
 L_PARENT_ID,
 L_LPN_ID
FROM DUAL;

/* INSERT INTO MTL_TRANSACTION_LOTS_INTERFACE */

 INSERT INTO MTL_TRANSACTION_LOTS_INTERFACE
 (TRANSACTION_INTERFACE_ID,
```

```
LAST_UPDATE_DATE,
  LAST_UPDATED_BY,
  CREATION_DATE,
  CREATED_BY,
  LAST_UPDATE_LOGIN,
  LOT_NUMBER,
  TRANSACTION_QUANTITY,
  PRIMARY_QUANTITY,
  SERIAL_TRANSACTION_TEMP_ID,
  PRODUCT_CODE,
  PRODUCT_TRANSACTION_ID
 )
 SELECT
  MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
  SYSDATE,
  1,
  SYSDATE,
  1,
  1,
  'LOT1', --THE SECOND LOT NUMBER
  2,
  2,
  MTL_MATERIAL_TRANSACTIONS_S.NEXTVAL,
  'RCV',
  PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
 FROM DUAL;


/* INSERT INTO MTL_SERIAL_NUMBERS_INTERFACE */

INSERT INTO MTL_SERIAL_NUMBERS_INTERFACE
(TRANSACTION_INTERFACE_ID,
 LAST_UPDATE_DATE,
 LAST_UPDATED_BY,
 CREATION_DATE,
 CREATED_BY,
 LAST_UPDATE_LOGIN,
 FM_SERIAL_NUMBER,
 TO_SERIAL_NUMBER,
 PRODUCT_CODE,
 PRODUCT_TRANSACTION_ID
)
SELECT
 MTL_MATERIAL_TRANSACTIONS_S.CURRVAL
 SYSDATE,
 1,
 SYSDATE,
 1,
 1,
 'SER3', --From Serial Number
 'SER4', --To Serial Number
 'RCV',
 PO.RCV_TRANSACTIONS_INTERFACE_S.CURRVAL --This should join to
RCV_TRANSACTIONS_INTERFACE.INTERFACE_TRANSACTION_ID
FROM DUAL;

COMMIT;

EXCEPTION
  WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('ERROR');
END;
/
```

Stuck Transaction Scenarios and Scripts

The following section contains information about reprocessing an errored row for an ASN/RECEIVE transaction. Run the following queries to identify the cause of the error, and fix the error before reprocessing any stuck records. After you fix the errors, check the following cases studies and run appropriate scripts.

> **Note:** Verify the Receiving transaction manager is not running while running the following script.

```
SELECT ERROR_MESSAGE, INTERFACE_LINE_ID, INTERFACE_HEADER_ID
FROM PO_INTERFACE_ERRORS
WHERE INTERFACE_LINE_ID IN
  (SELECT INTERFACE_TRANSACTION_ID
   FROM RCV_TRANSACTIONS_INTERFACE
   WHERE PROCESSING_STATUS_CODE IN('PENDING', 'ERROR', 'RUNNING',
'COMPLETED'));

SELECT ERROR_MESSAGE, INTERFACE_LINE_ID, INTERFACE_HEADER_ID
FROM PO_INTERFACE_ERRORS
WHERE INTERFACE_HEADER_ID IN
  (SELECT INTERFACE_HEADER_ID
   FROM RCV_HEADERS_INTERFACE
   WHERE PROCESSING_STATUS_CODE IN('PENDING', 'ERROR', 'RUNNING'));
```

Case 1

Some of the rows in the RCV_TRANSACTIONS_INTERFACE for a given RCV_HEADERS_INTERFACE processed successfully. This means that the system created the shipment header in RCV_SHIPMENT_HEADERS, and one or more shipment lines exist in RCV_SHIPMENT_LINES. You can create a new header to process the stuck records. Reprocess the records as follows:

1. Create a backup of the RCV_TRANSACTIONS_INTERFACE records.

```
CREATE TABLE CASE1_RTI_BACKUP AS
SELECT RTI.* -- SELECT FOR STUCK RECORDS IN ONLINE/IMMEDIATE
FROM APPS.RCV_TRANSACTIONS_INTERFACE RTI,
  APPS.PO_LINE_LOCATIONS_ALL PLL,
  APPS.PO_HEADERS_ALL POH,
  APPS.RCV_HEADERS_INTERFACE RHI,
  APPS.RCV_SHIPMENT_HEADERS RSH
WHERE RTI.PO_HEADER_ID = POH.PO_HEADER_ID
 AND PLL.LINE_LOCATION_ID = RTI.PO_LINE_LOCATION_ID
 AND NVL(PLL.CLOSED_CODE,   'OPEN') = 'OPEN'
 AND RTI.SHIPMENT_HEADER_ID IS NOT NULL
 AND RTI.VENDOR_ID IS NOT NULL
 AND RTI.PROCESSING_MODE_CODE IN('ONLINE',   'IMMEDIATE',   'BATCH')
 AND RTI.PROCESSING_STATUS_CODE IN('PENDING',   'ERROR',
'RUNNING',   'COMPLETED')
 AND RSH.SHIPMENT_HEADER_ID = RTI.SHIPMENT_HEADER_ID
 AND EXISTS
  (SELECT 'ONE OR MORE SHIPMENT LINE(S) EXISTS'
   FROM APPS.RCV_SHIPMENT_LINES RSL
   WHERE RSL.SHIPMENT_HEADER_ID = RSH.SHIPMENT_HEADER_ID);
```

2. Create a backup of the RCV_HEADERS_INTERFACE records.

```
CREATE TABLE CASE1_RHI_BACKUP AS
SELECT RHI.*
FROM RCV_HEADERS_INTERFACE RHI
WHERE HEADER_INTERFACE_ID IN
  (SELECT DISTINCT HEADER_INTERFACE_ID
   FROM CASE1_RTI_BACKUP);
```

3. Some records might have been ASN direct delivery for example, RTI.TRANSACTION_TYPE = SHIP and RTI.AUTO_TRANSACT_CODE = DELIVER. In this case, receiving changes the TRANSACTION_TYPE to RECEIVE. When the rows are reprocessed, they reverted back to their original state.

```
DECLARE
CURSOR C1 IS
SELECT RHI.HEADER_INTERFACE_ID
FROM CASE2_RHI_BACKUP RHI,
CASE2_RTI_BACKUP RTI
WHERE RHI.HEADER_INTERFACE_ID = RTI.HEADER_INTERFACE_ID
AND RHI.ASN_TYPE IN ('ASN','ASBN')
AND RTI.TRANSACTION_TYPE = 'RECEIVE';

BEGIN
FOR REC1 IN C1
LOOP

UPDATE RCV_TRANSACTIONS_INTERFACE
SET TRANSACTION_TYPE = 'SHIP',
SHIPMENT_HEADER_ID = NULL,
SHIPMENT_LINE_ID = NULL
WHERE HEADER_INTERFACE_ID = REC1.HEADER_INTERFACE_ID;

END LOOP;

END ;
/
```

4. Update the records to be processed.

```
DECLARE
CURSOR C1 IS
SELECT *
FROM   CASE1_RHI_BACKUP;

CURSOR C2 IS
SELECT *
FROM   CASE1_RTI_BACKUP;

BEGIN
FOR REC1 IN C1
LOOP

UPDATE RCV_HEADERS_INTERFACE
SET   PROCESSING_STATUS_CODE='PENDING',
      RECEIPT_HEADER_ID=NULL,
      PROCESSING_REQUEST_ID=NULL,
      VALIDATION_FLAG='Y', --'Y' FOR PRE-PROCESSOR DERIVATION.
      EXPECTED_RECEIPT_DATE=SYSDATE,
      SHIPPED_DATE=SYSDATE,
      SHIPMENT_NUM='&SHIPMENT_NUM' --NEW SHIPMENT NUMBER
WHERE HEADER_INTERFACE_ID = REC1.HEADER_INTERFACE_ID;

END LOOP;

FOR REC2 IN C2
LOOP

UPDATE RCV_TRANSACTIONS_INTERFACE
   SET PROCESSING_STATUS_CODE='PENDING',
       PROCESSING_MODE_CODE='BATCH',
       TRANSACTION_STATUS_CODE='PENDING',
       PROCESSING_REQUEST_ID=NULL,
       VALIDATION_FLAG='Y', --'Y' FOR PRE-PROCESSOR DERIVATION.
       SHIPMENT_HEADER_ID=NULL,
       TRANSACTION_DATE=SYSDATE, --THIS IS REQUIRED IF THE RECORD
WAS STUCK SOMETIME BACK
       CREATION_DATE=SYSDATE, --AND THE CORRESPONDING PERIODS ARE
NOW CLOSED
       EXPECTED_RECEIPT_DATE=SYSDATE
WHERE  INTERFACE_TRANSACTION_ID = REC2.INTERFACE_TRANSACTION_ID;

END LOOP;

END;
/

COMMIT;
```

5. Run the processor.

Case 2

RCV_HEADERS_INTERFACE processed successfully. This means that the shipment header is created in RCV_SHIPMENT_HEADERS but none of the RCV_TRANSACTIONS_INTERFACEs for the given RCV_HEADERS_INTERFACE

have processed yet. This means that no shipment lines exist in RCV_SHIPMENT_LINES. To reprocess records, delete the orphan RCV_SHIPMENT_HEADERS record, and reset the RCV_HEADERS_INTERFACE and the RCV_TRANSACTIONS_INTERFACE.

1. Create a backup of the RCV_TRANSACTIONS_INTERFACE, RCV_HEADERS_INTERFACE, and RCV_SHIPMENT_LINES records tables.

```
CREATE TABLE CASE2_RTI_BACKUP AS
SELECT RTI.* -- SELECT FOR STUCK RECORDS IN ONLINE/IMMEDIATE
FROM APPS.RCV_TRANSACTIONS_INTERFACE RTI,
  APPS.PO_LINE_LOCATIONS_ALL PLL,
  APPS.PO_HEADERS_ALL POH,
  APPS.RCV_SHIPMENT_HEADERS RSH
WHERE RTI.PO_HEADER_ID = POH.PO_HEADER_ID
 AND PLL.LINE_LOCATION_ID = RTI.PO_LINE_LOCATION_ID
 AND NVL(PLL.CLOSED_CODE,   'OPEN') = 'OPEN'
 AND RTI.SHIPMENT_HEADER_ID IS NOT NULL
 AND RTI.VENDOR_ID IS NOT NULL
 AND RTI.PROCESSING_MODE_CODE IN('ONLINE',   'IMMEDIATE',   'BATCH')
 AND RTI.PROCESSING_STATUS_CODE IN('PENDING',   'ERROR',
'RUNNING',   'COMPLETED')
 AND RSH.SHIPMENT_HEADER_ID = RTI.SHIPMENT_HEADER_ID
 AND NOT EXISTS
  (SELECT 'NO SHIPMENT LINES'
   FROM APPS.RCV_SHIPMENT_LINES RSL
   WHERE RSL.SHIPMENT_HEADER_ID = RSH.SHIPMENT_HEADER_ID);

CREATE TABLE CASE2_RHI_BACKUP AS
SELECT RHI.*
FROM RCV_HEADERS_INTERFACE RHI
WHERE HEADER_INTERFACE_ID IN
  (SELECT DISTINCT HEADER_INTERFACE_ID
   FROM CASE2_RTI_BACKUP);

CREATE TABLE CASE2_RSH_BACKUP AS
SELECT *
FROM RCV_SHIPMENT_HEADERS
WHERE SHIPMENT_HEADER_ID IN
  (SELECT SHIPMENT_HEADER_ID
   FROM CASE2_RTI_BACKUP);
```

2. Delete the orphan RCV_SHIPMENT_LINES records.

```
DELETE FROM RCV_SHIPMENT_HEADERS
WHERE SHIPMENT_HEADER_ID IN
  (SELECT SHIPMENT_HEADER_ID
   FROM CASE2_RTI_BACKUP);
```

3. Some records might have been ASN direct delivery (for example, RTI.TRANSACTION_TYPE = SHIP and RTI.AUTO_TRANSACT_CODE = DELIVER). In this case, receiving changes the TRANSACTION_TYPE to RECEIVE. When you reprocess the row, it needs to revert back to the original state.

```
DECLARE
CURSOR C1 IS
SELECT RHI.HEADER_INTERFACE_ID
FROM CASE2_RHI_BACKUP RHI,
CASE2_RTI_BACKUP RTI
WHERE RHI.HEADER_INTERFACE_ID =RTI.HEADER_INTERFACE_ID
AND RHI.ASN_TYPE IN ('ASN','ASBN')
AND RTI.TRANSACTION_TYPE = 'RECEIVE';

BEGIN
FOR REC1 IN C1
LOOP

UPDATE RCV_TRANSACTIONS_INTERFACE
SET TRANSACTION_TYPE = 'SHIP',
SHIPMENT_HEADER_ID = NULL
WHERE HEADER_INTERFACE_ID = REC1.HEADER_INTERFACE_ID;

END LOOP;

END ;
/
```

4. Update the records to be processed.

```
DECLARE
CURSOR C1 IS
SELECT *
FROM   CASE2_RHI_BACKUP;

CURSOR C2 IS
SELECT *
FROM   CASE2_RTI_BACKUP;

BEGIN
FOR REC1 IN C1
LOOP

UPDATE RCV_HEADERS_INTERFACE
SET   PROCESSING_STATUS_CODE='PENDING',
      RECEIPT_HEADER_ID=NULL,
      PROCESSING_REQUEST_ID=NULL,
      VALIDATION_FLAG='Y',             -- 'Y' FOR PRE-PROCESSOR
DERIVATION.
      EXPECTED_RECEIPT_DATE=SYSDATE,
      SHIPPED_DATE=SYSDATE,
WHERE HEADER_INTERFACE_ID = REC1.HEADER_INTERFACE_ID;

END LOOP;

FOR REC2 IN C2
LOOP

UPDATE RCV_TRANSACTIONS_INTERFACE
   SET PROCESSING_STATUS_CODE='PENDING',
       PROCESSING_MODE_CODE='BATCH',
       TRANSACTION_STATUS_CODE='PENDING',
       PROCESSING_REQUEST_ID=NULL,
       VALIDATION_FLAG='Y',              -- 'Y' FOR PRE-PROCESSOR
DERIVATION.
       SHIPMENT_HEADER_ID=NULL,
       TRANSACTION_DATE=SYSDATE,         -- THIS IS REQUIRED IF THE
RECORD WAS STUCK SOMETIME BACK
       CREATION_DATE=SYSDATE,            -- AND THE CORRESPONDING
PERIODS ARE NOW CLOSED
       EXPECTED_RECEIPT_DATE=SYSDATE
WHERE  INTERFACE_TRANSACTION_ID = REC2.INTERFACE_TRANSACTION_ID;

END LOOP;

END;
/

COMMIT;
```

5. Run the processor.

Case 3

Reprocess the interorg shipments or internal requisition documents that are stuck in the RCV_TRANSACTIONS_INTERFACE and run the Receiving Transaction processor.

```
CREATE TABLE RTI_BACKUP AS
SELECT RTI.*
FROM APPS.RCV_TRANSACTIONS_INTERFACE RTI
WHERE SHIPMENT_HEADER_ID IN (
  SELECT DISTINCT RTI.SHIPMENT_HEADER_ID
  FROM APPS.RCV_TRANSACTIONS_INTERFACE RTI,
  APPS.RCV_SHIPMENT_HEADERS RSH
  WHERE RTI.SHIPMENT_HEADER_ID = RSH.SHIPMENT_HEADER_ID
  AND RSH.RECEIPT_SOURCE_CODE IN ('INVENTORY','INTERNAL ORDER')
  AND RTI.PROCESSING_STATUS_CODE NOT IN ('PENDING','RUNNING')
   );

CREATE TABLE RHI_BACKUP AS
 SELECT RHI.*
 FROM APPS.RCV_HEADERS_INTERFACE RHI
 WHERE HEADER_INTERFACE_ID IN (SELECT DISTINCT HEADER_INTERFACE_ID
       FROM RTI_BACKUP);
```

```
DECLARE
 CURSOR C1 IS
 SELECT *
 FROM  RHI_BACKUP;
 .
 CURSOR C2 IS
 SELECT *
 FROM   RTI_BACKUP;
 .
 BEGIN
 FOR REC1 IN C1
 LOOP

 UPDATE APPS.RCV_HEADERS_INTERFACE
 SET   PROCESSING_STATUS_CODE='PENDING',
       RECEIPT_HEADER_ID=NULL,
       PROCESSING_REQUEST_ID=NULL,
       VALIDATION_FLAG='Y',
       EXPECTED_RECEIPT_DATE=SYSDATE,
       SHIPPED_DATE=SYSDATE
 WHERE HEADER_INTERFACE_ID = REC1.HEADER_INTERFACE_ID;
 .
 .
 END LOOP;
 .
 FOR REC2 IN C2
 LOOP
 .
 UPDATE APPS.RCV_TRANSACTIONS_INTERFACE
    SET PROCESSING_STATUS_CODE='PENDING',
        PROCESSING_MODE_CODE='BATCH',
        TRANSACTION_STATUS_CODE='PENDING',
        PROCESSING_REQUEST_ID=NULL,
        VALIDATION_FLAG='Y',
        TRANSACTION_DATE=SYSDATE,
        CREATION_DATE=SYSDATE,
        EXPECTED_RECEIPT_DATE=SYSDATE ,
        INTERFACE_TRANSACTION_QTY = NULL
 WHERE  INTERFACE_TRANSACTION_ID = REC2.INTERFACE_TRANSACTION_ID;
 .
 UPDATE APPS.RCV_SHIPMENT_HEADERS
 SET RECEIPT_NUM = NULL
 WHERE SHIPMENT_HEADER_ID = REC2.SHIPMENT_HEADER_ID;
 .
 END LOOP;
 .
 END;
 /
 .
 COMMIT
```

Case 4

Reprocess the transactions that are stuck in RCV_TRANSACTIONS_INTERFACE that do not have any header information, such as DELIVER, RETURNS, CORRECTIONS, and INSPECTION. When finished, run the Receiving Transaction processor.

```
CREATE TABLE RTI_BACKUP AS
SELECT RTI.*
FROM APPS.RCV_TRANSACTIONS_INTERFACE RTI
    WHERE RTI.PROCESSING_STATUS_CODE NOT IN ('PENDING','RUNNING')
    AND RTI.HEADER_INTERFACE_ID IS NULL;

UPDATE APPS.RCV_TRANSACTIONS_INTERFACE
  SET PROCESSING_STATUS_CODE='PENDING',
        PROCESSING_MODE_CODE='BATCH',
        TRANSACTION_STATUS_CODE='PENDING',
        PROCESSING_REQUEST_ID=NULL,
        VALIDATION_FLAG='Y',
        TRANSACTION_DATE=SYSDATE,
        CREATION_DATE=SYSDATE,
        EXPECTED_RECEIPT_DATE=SYSDATE ,
        INTERFACE_TRANSACTION_QTY = NULL
  WHERE  RTI.PROCESSING_STATUS_CODE NOT IN ('PENDING','RUNNING')
      AND RTI.HEADER_INTERFACE_ID IS NULL;

UPDATE APPS.RCV_TRANSACTIONS_INTERFACE
  SET PROCESSING_STATUS_CODE='PENDING',
        PROCESSING_MODE_CODE='BATCH',
        TRANSACTION_STATUS_CODE='PENDING',
        PROCESSING_REQUEST_ID=NULL,
        VALIDATION_FLAG='Y',
        TRANSACTION_DATE=SYSDATE,
        CREATION_DATE=SYSDATE,
        EXPECTED_RECEIPT_DATE=SYSDATE ,
        INTERFACE_TRANSACTION_QTY = NULL
  WHERE  RTI.PROCESSING_STATUS_CODE NOT IN ('PENDING','RUNNING')
      AND RTI.HEADER_INTERFACE_ID IS NULL;
```

Oracle Warehouse Management LPN Interface Table Supplemental Information

The Receiving Transaction Manager uses the WMS_LPN_INTERFACE to process receiving transactions. The Receiving Transaction Manager uses it to store LPN transaction information. After the Receiving Transaction Manager processes a transaction, it stores the LPN information in the WMS_LICENSE_PLATE_NUMBERS table. The WMS_LPN_INTERFACE table contains the same columns as the WMS_LICENSE_PLATE_NUMBERS table as well as additional columns that the Receiving Transaction Manager uses.

Setting Up the LPN Interface Table

The system uses the WMS_LPN_INTERFACE table in conjunction with other receiving interfaces when processing receiving transactions. For example, if you are packing LPN1 into LPN2, then the system inserts the following into the WMS_LPN_INTERFACE table.

LPN	PARENT_LPN	SOURCE_GROUP_ID
LPN1	LPN2	<RTI.LPN_GROUP_ID>

If you unpack LPN1 into LPN2, then the system inserts the following in to the WMS_LPN_INTERFACE table.

LPN	PARENT_LPN	SOURCE_GROUP_ID
LPN1	<NULL>	<RTI.LPN_GROUP_ID>

For transactions that involve a new LPN, such as ASN creation, you insert the following in to the WMS_INTERFACE table. The system later transfers this information to the WMS_LICENSE_PLATE_NUMBERS table.

LPN	PARENT_LPN	Weight	Attribute1	Attribute2	Attribute3	SOURCE_GROUP_ID
LPN2	<NULL>	400	sdfds	efgsdf	xyzsdf	<RTI.LPN_GROUP_ID>
LPN1	LPN1	200	abc	efg	xyz	<RTI.LPN_GROUP_ID>

Validation

The Receiving Transaction Manager validates the LPN and Transfer LPN columns stamped on the Receiving Transactions Interface. The system does not perform an actual validation on the WMS_LPN_INTERFACE table.

Related Topics

Oracle Warehouse Management License Plate Number Interface Table Description, *Oracle Manufacturing APIs and Open Interfaces Manual*

17

eAM Open Interfaces and APIs

This chapter covers the following topics:

- eAM Open Interfaces and APIs
- eAM Item Open Interface
- eAM Asset Number Open Interface
- eAM Asset Genealogy Open Interface
- eAM Meter Reading Open Interface
- Asset Number API
- Asset Attribute Values API
- Asset Attribute Groups API
- Asset Routes API
- Asset Areas API
- Department Approvers API
- EAM Parameters API
- EAM Meters API
- EAM Meter Association API
- Meter Reading API
- EAM PM Schedules API (includes PM Rules as children records)
- Activity Creation API
- EAM Activity Association API
- EAM Activity Suppression API
- EAM Set Name API
- Maintenance Object Instantiation API
- Work Order Business Object API

- Work Request API

eAM Open Interfaces and APIs

Oracle Enterprise Asset Management provides several open interfaces and APIs, enabling you to link with non-Oracle applications, applications you build, and applications on other computers.

This section includes the following topics:

eAM Item Open Interface

The eAM Item Open Interface enables you to import Asset Groups, Activities, and Rebuildable Items into the eAM application, using a batch process. Create them as new items or update existing items (Asset Groups, Activities, or Rebuildable Items). The Item Open Interface validates your data, ensuring that your imported items contain the same item detail as items you enter manually within the Master Item window.

You can import item category assignments. You can item category assignments simultaneously with an importing items process, or as a separate task.

When importing items through the Item Open Interface, new items are created in your item master organization, existing items are updated, or existing items are assigned to additional organizations. You can specify values for all item attributes, or you can specify just a few attributes and let the remainder default or remain null. The Item Open Interface enables import revision details, including past and future revisions and effectivity dates. Validation of imported items is performed using the same rules as the item definition windows to ensure valid items. See: Overview of Engineering Prototype Environment, *Oracle Engineering User's Guide* and Defining Items, *Oracle Inventory User's Guide*.

The Item Open Interface reads data from three tables for importing items and item details. Use the MTL_SYSTEM_ITEMS_INTERFACE table for new item numbers and all item attributes. This is the main item interface table and may be the only table you choose to use. When importing revision details for new items, use the MTL_ITEM_REVISIONS_INTERFACE table. This table is used for revision information and is not required. When importing item category assignments, use the MTL_ITEM_CATEGORIES_INTERFACE table to store data about item assignments to category sets and categories to import into the Oracle Inventory MTL_ITEM_CATEGORIES table. A fourth table, MTL_INTERFACE_ERRORS, is used for error tracking of all items that the Item Interface fails.

Before using the Item Open Interface, write and execute a custom program that extracts item information from your source system and inserts the records into the MTL_SYSTEM_ITEMS_INTERFACE and (if revision detail is included) MTL_ITEMS_REVISIONS_INTERFACE, and MTL_ITEM_CATEGORIES_INTERFACE tables. After loading item, revision, and item category assignment records into these interface tables, execute the Item Open Interface to import the data. The Item Open Interface assigns defaults, validates included data, and then imports the new items.

Note: Import items into a master organization before importing items

into additional organizations. Specify only your master organization on a first pass execution. After this completes, you can execute the Item Open Interface again, this time specifying an additional or all organizations.

Setting up the item open interface:

1. Create Indexes for Performance. Create the following indexes to improve the Item Open Interface performance.

   First, determine which segments are enabled for the System Items flexfield.

   Next, for example, if you have a two-segment flexfield, with SEGMENT8 and SEGMENT12 enabled:

   ```
   SQL> create unique index MTL_SYSTEM_ITEMS_B_UC1 on
   MTL_SYSTEM_ITEMS_B (ORGANIZATION_ID, SEGMENT8, SEGMENT12);SQL>
   create unique index MTL_SYSTEM_ITEMS_INTERFACE_UC1 on
   mtl_system_items_interface (organization_id, segment8, segment12);
   ```

   If you plan to populate the ITEM_NUMBER column in mtl_system_items_interface instead of the item segment columns, do not create the MTL_SYSTEM_ITEMS_INTERFACE_UC1 unique index. Instead, create MTL_SYSTEM_ITEMS_INTERFACE_NC1 non-unique index on the same columns.

   Create the following indexes for optimum performance:

   MTL_SYSTEM_ITEMS_B

   - Non-Unique Index on organization_id, segment*n*
   - You need at least one indexed, mandatory segment.

   MTL_SYSTEM_ITEMS_INTERFACE

   - Non Unique Index on inventory_item_id, organization_id
   - Non Unique Index on Item _number
   - Unique Index on Transaction_id

     Note: Recreate this Index as Non Unique, if you are populating organization_code instead of organization_id; it includes the previously enabled segment(s) for the System Item Key Flexfield. Use the created default index if you are using segment1.

   MTL_ITEM_REVISIONS_INTERFACE

   - Non Unique Index on set_process_id
   - Non Unique Index on Transaction_id

- Non Unique Index on Organization_id, Inventory_item_id, Revision

MTL_ITEM_CATEGORIES_INTERFACE

- Non Unique Index on inventory_item_id, category_id
- Non Unique Index on set_process_id

> **Tip:** Populate _id fields whenever possible. Populating inventory_item_id instead of segment (n) for Update Mode improves performance. Populating organization_id, instead of organization_code for both Create and Update modes, reduces processing time.

2. Start the Concurrent Manager.

   Because the Item Open Interface process is launched and managed via the concurrent manager, ensure that the concurrent manager is executing before importing any items.

3. Set Profile Option Defaults.

   Some columns use profile options as default values. Set these profiles, if you want them to default. See: Inventory Profile Options, *Oracle Inventory User's Guide* and Overview of Inventory Setup, *Oracle Inventory User's Guide*.

Execution Steps:

1. Populate the interface tables.

   The item interface table MTL_SYSTEM_ITEMS_INTERFACE contains *every* column in the Oracle Inventory item master table, MTL_SYSTEM_ITEMS. The columns in the item interface correspond directly to those in the item master table. Except for ITEM_NUMBER or SEGMENT*n* columns, ORGANIZATION_CODE or ORGANIZATION_ID, DESCRIPTION, PROCESS_FLAG, and TRANSACTION_TYPE, all of these columns are optional, either because they have defaults that can derive, or because the corresponding attributes are optional and may be null.

   You may put in details about other interface tables not used by the Item Open Interface.

   Currently, the interface does not support the MTL_CROSS_REFERENCE_INTERFACE or MTL_SECONDARY_LOCS_INTERFACE.

   The MTL_ITEM_CATEGORIES_INTERFACE is used by the Item Open Interface for both internal processing of default category assignments, *and* to retrieve data populated by the user to be imported into the Oracle Inventory

MTL_ITEM_CATEGORIES table.

Column Names (partial list of columns)	Instruction
PROCESS_FLAG	Enter 1 for pending data to be imported. After running the import process, the PROCESS_FLAG of the corresponding rows will be set to different values, indicating the results of the import (1 = Pending, 2 = Assign complete, 3 = Assign/validation failed, 4 = Validation succeeded; import failed, 5 = Import in process, 6 = Import succeeded)
TRANSACTION_TYPE	Enter CREATE to create a new item, or UPDATE to update existing items.
SET_PROCESS_ID	Enter an arbitrary number. Rows designated with the same value for SET_PROCESS_ID will process together.
ORGANIZATION_CODE	Enter the organization that the new item will import into.
SEGMENT1~20	Corresponds to the item name (for example, the name of the Asset Group, Asset Activity, or Rebuildable Item)
DESCRIPTION	Enter the description of the item.
EAM_ITEM_TYPE	Enter 1 for Asset Group, 2 for Asset Activity, or 3 for Rebuildable Item.
INVENTORY_ITEM_FLAG	Enter Y for eAM items.
MTL_TRANSACTIONS_ENABLED_FLAG	Enter N for eAM items.
EFFECTIVITY_CONTROL	Enter 2 for Unit Effectivity Control for eAM Asset Groups.
SERIAL_NUMBER_CONTROL_CODE	Enter 2 (Predefined) for Asset Groups; this should be NULL for Asset Activities.

Column Names (partial list of columns)	Instruction
AUTO_SERIAL_ALPHA_PREFIX	Serial Number Prefix
START_AUTO_SERIAL_NUMBER	Start Serial Number

Note: For information about columns not discussed, see Table and View Definitions, *Oracle Inventory Technical Reference Manual*.

2. Launch the Item Import process.

   1. Navigate to the Import Items window.

      Choose an organization, if you have not specified one already. Import items into the master organization before importing them to additional children organization.

   2. Enter parameters:

Parameter	Description
All Organizations	Select Yes to run the interface for all organization codes within the item interface table. Select No to run the interface for only interface rows within the current organization.
Validate Items	Select Yes to validate all items, and their information residing in the interface table, that have not yet been validated. If items are not validated, they will not be imported into eAM. Select No to not validate items within the interface table. Use this option if you have previously run the Item Open interface and responded Yes in the Validate Items parameter, and No in the Process Items parameter, and now need to process your items.

Parameter	Description
Process Items	Select Yes to import all qualifying items in the interface table into eAM. Select No to not import items into eAM. Use this option, along with Yes in the Delete Processed Rows parameter, to remove successfully processed rows from the interface table, without performing any other processing. You can also use this option, with Yes in the Validate Items parameter, to validate items without any processing.
Delete Processed Rows	Select Yes to delete successfully processed rows from the item interface tables. Select No to leave all rows in the item interface tables.
Process Set	Enter a set id number for the set of rows in the interface table to process. The interface process will process rows having that id in the SET_PROCESS_ID column. If you leave this parameter blank, all rows are picked up for processing, regardless of the SET_PROCESS_ID column value.
Create or Update Items	Select 1 to create new items. Select 2 to update existing items. You can create or update items via separate executions of the Import Items process.

3. Choose OK.
4. Choose Submit to launch the Import Asset Number process. You can view its progress by choosing View from the tool bar, and then selecting Requests.

Related Topics

Oracle Manufacturing APIs and Open Interfaces Manual, Release 11*i*

Defining Asset Groups, *Oracle Enterprise Asset Management User's Guide*

Defining Activities, *Oracle Enterprise Asset Management User's Guide*

Defining Rebuildable Items, *Oracle Enterprise Asset Management User's Guide*

eAM Asset Number Open Interface

The eAM Asset Number Open Interface enables you to import Asset Numbers into eAM, using a batch process. Optionally import Asset Number attributes. You can create new Asset Numbers and attributes, or update existing Asset Numbers and attributes.

Execution Steps:

1. Populate the interface tables with the import information.

   The two item interface tables to populate are MTL_EAM_ASSET_NUM_INTERFACE (MEANI), and the MTL_EAM_ATTR_VAL_INTERFACE (MEAVI). The MTL_EAM_ASSET_NUM_INTERFACE table stores relevant Asset Number information. If the asset's attributes are also imported, that information is stored in the MTL_EAM_ATTR_VAL_INTERFACE.

Column Name (partial list of columns)	Instruction
BATCH_ID	Enter an arbitrary number. Rows designated with the same BATCH_ID will process together.
PROCESS_FLAG	Enter a P for pending. This value will change to S if the import is successful, or E if the row contains an error.
IMPORT_MODE	Enter 0 to create new rows (asset numbers), or 1 to update existing rows.
IMPORT_SCOPE	Enter 0 to import both Asset Numbers and Attributes, 1 to import Asset Numbers only, or 2 to import Attributes only.
INVENTORY_ITEM_ID	Enter the Asset Group to associate with the imported Asset Number(s).
SERIAL_NUMBER	Enter the name of the Asset Number.
ORGANIZATION_CODE	Enter the current organization.

Column Name (partial list of columns)	Instruction
OWNING_DEPARTMENT_CODE	Enter the Owning Department of the asset number(s).
ERROR_CODE	This column will update by the Import process if an error occurs.
ERROR_MESSAGE	This column will update by the Import process if an error occurs.
INTERFACE_HEADER_ID	This is used with the identically named column in the MEAVI table, to relate the Attributes associated with an Asset Number.

Column Name (partial list of columns)	Instruction
PROCESS_STATUS	Enter P (Pending). This value will change to S if the import is successful, or E if the row contains an error.
INTERFACE_HEADER_ID	Foreign key of the identically named column in the MEANI table to relate to the Asset Number an Attribute is associated with.
INTERFACE_LINE_ID	A unique key
END_USER_COLUMN_NAME	Corresponds with the Attribute Name
ATTRIBUTE_CATEGORY	Corresponds with the Attribute Group
LINE_TYPE	Enter 1 if the Attribute is of type VARCHAR2, 2 if it is of type NUMBER, or 3 if it is of type DATE.
ATTRIBUTE_VARCHAR2_VALUE	Value of the Attribute; used with LINE_TYPE = 1
ATTRIBUTE_NUMBER_VALUE	Value of the Attribute; used with LINE_TYPE = 2

Column Name (partial list of columns)	Instruction
ATTRIBUTE_DATE_VALUE	Value of the Attribute; used with LINE_TYPE = 3
ERROR_NUMBER	This column will update by the Import process if an error occurs.
ERROR_MESSAGE	This column will update by the Import process if an error occurs.

Note: For information about columns not discussed, see Table and View Definitions, *Oracle Enterprise Asset Management Technical Reference Manual.*

2. Launch the Asset Number Import process to import interface information into the MTL_SERIAL_NUMBERS production table.

   1. Navigate to the Asset Number Import window.

      Choose an organization if you have not specified one already. Import Asset Numbers into the master organization before importing them into additional children organizations.

   2. Enter parameters:

Parameter	Description
Batch ID	This is the same value that is populated in the BATCH_ID column within the MEANI table.
Purge Option	Select Yes to delete rows in the interface tables after they have successfully imported into the production tables. Select No to keep all rows in the interface tables after they have successfully imported into the production tables. Any failed rows with error messages will not delete.

3. Choose OK.

4. Choose Submit to launch the Asset Number Import process. You can view its progress by choosing View from the tool bar, and then selecting Requests.

Create and Update Asset Number API:

The MTL_EAM_ASSET_NUMBER_PUB public API is used to create and update Asset Numbers in the MTL_SERIAL_NUMBERS table, and is called from the Asset Number Open Interface process.

Column Name	Type	Required	Optional	Default
P_API_VERSION	Number	Yes	-	-
P_INIT_MSG_LIST	Varchar2	-	Yes	FND_API.G_FALSE
P_COMMIT	Varchar2	-	Yes	FND_API.G_FALSE
P_VALIDATION_LEVEL	Number	-	Yes	FND_API.G_VALID_LEVEL_FULL
P_INVENTORY_ITEM_ID	Number	-	-	-
P_SERIAL_NUMBER	Varchar2(30)	-	-	-
P_INITIALIZATION_DATE	Date	-	-	-
P_DESCRIPTIVE_TEXT	Varchar2(240)	-	-	-
P_CURRENT_ORGANIZATION_ID	Number	-	-	-
P_ATTRIBUTE CATEGORY	Varchar2(30)	-	-	-

Column Name	Type	Required	Optional	Default
P_ATTRIBUTE1 ~15	Varchar2(150)	-	-	-
P_GEN_OBJECT _ID	Number	-	-	-
P_CATEGORY_I D	Number	-	-	-
P_WIP_ACCOU NTING_CLASS_ CODE	Varchar2(10)	-	-	-
P_MAINTAINA BLE_FLAG	Varchar2(1)	-	-	-
P_OWNING_DE PARTMENT_ID	Number	-	-	-
P_DEPENDENT _ASSET_FLAG	Varchar2(1)	-	-	-
P_NETWORK_A SSET_FLAG	Varchar2(1)	-	-	-
P_FA_ASSET_ID	Number	-	-	-
P_PN_LOCATIO N_ID	Number	-	-	-
P_EAM_LOCAT ION_ID	Number	-	-	-
P_ASSET_STAT US_CODE	Varchar2(30)	-	-	-
P_ASSET_CRITI CALITY_CODE	Varchar2(30)	-	-	-
P_LAST_UPDA TE_DATE	Date	Yes	-	-

Column Name	Type	Required	Optional	Default
P_LAST_UPDATED_BY	Number	Yes	-	-
P_CREATION_DATE	Date	Yes	-	-
P_CREATED_BY	Number	Yes	-	-
P_LAST_UPDATE_LOGIN	Number	Yes	-	-
P_REQUEST_ID	Number	-	Yes	-
P_PROGRAM_APPLICATION_ID	Number	-	Yes	-
P_PROGRAM_ID	Number	-	Yes	-
P_PROGRAM_UPDATE_DATE	Date	-	Yes	-

Create and Update Extensible Asset Attribute Values API:

The MTL_EAM_ASSET_ATTR_VALUES_PUB public API is used to create and update asset extensible attribute values, and is called from the Asset Extensible Attributes Open Interface process. This interface process is used when the IMPORT_SCOPE column is set to 0 or 2 within the MTL_EAM_ASSET_NUM_INTERFACE table.

Column Name	Type	Required	Optional	Default
P_API_VERSION	Number	Yes	-	-
P_INIT_MSG_LIST	Varchar2	-	Yes	FND_API.G_FALSE
P_COMMIT	Varchar2	-	Yes	FND_API.G_FALSE

Column Name	Type	Required	Optional	Default
P_VALIDATION_LEVEL	Number	-	Yes	FND_API.G_VALID_LEVEL_FULL
P_ATTRIBUTE CATEGORY	Varchar2(30)	-	-	-
P_C_ATTRIBUTE1~20	Varchar2(150)	-	-	-
P_N_ATTRIBUTE1~10	Number	-	-	-
P_D_ATTRIBUTE1~10	Date	-	-	-
P_LAST_UPDATE_DATE	Date	Yes	-	-
P_LAST_UPDATED_BY	Number	Yes	-	-
P_CREATION_DATE	Date	Yes	-	-
P_CREATED_BY	Number	Yes	-	-
P_LAST_UPDATE_LOGIN	Number	Yes	-	-
P_REQUEST_ID	Number	-	Yes	-
P_PROGRAM_APPLICATION_ID	Number	-	Yes	-
P_PROGRAM_ID	Number	-	Yes	-
P_PROGRAM_UPDATE_DATE	Date	-	Yes	-

Create and Update Asset Genealogy and Hierarchy API:

The INV_GENEALOGY_PUB public API is used to create and update asset genealogy and hierarchy information in the MTL_OBJECT_GENEALOGY table, and is called from the Asset Number Open Interface process.

Column Name	Type	Required	Default
P_API_VERSION	Number	Yes	-
P_INIT_MSG_LIST	Varchar2	-	FND_API.G_FALSE
P_COMMIT	Varchar2	-	FND_API.G_FALSE
P_VALIDATION_LEVEL	Number	-	FND_API.G_VALID_LEVEL_FULL
P_OBJECT_TYPE	Number	Yes	-
P_PARENT_OBJECT_TYPE	Number	-	-
P_OBJECT_ID	Number	-	-
P_OBJECT_NUMBER	Varchar2	-	-
P_INVENTORY_ITEM_ID	Number	-	-
P_ORG_ID	Number	-	-
P_PARENT_OBJECT_ID	Number	-	-
P_PARENT_OBJECT_NUMBER	Varchar2	-	-
P_PARENT_INVENTORY_ITEM_ID	Number	-	-
P_PARENT_ORG_ID	Number	-	-
P_GENEALOGY_ORIGIN	Number	-	-

Column Name	Type	Required	Default
P_GENEALOGY_TYPE	Number	-	-
P_START_DATE_ACTIVE	Date	-	-
P_END_DATE_ACTIVE	Date	-	-
P_ORIGIN_TXN_ID	Number	-	-
P_UPDATE_TXN_ID	Number	-	-
P_LAST_UPDATE_DATE	Date	Yes	-
P_LAST_UPDATED_BY	Number	Yes	-
P_CREATION_DATE	Date	Yes	-
P_CREATED_BY	Number	Yes	-
P_LAST_UPDATE_LOGIN	Number	Yes	-
P_REQUEST_ID	Number	-	-
P_PROGRAM_APPLICATION_ID	Number	-	-
P_PROGRAM_ID	Number	-	-
P_PROGRAM_UPDATE_DATE	Date	-	-

Related Topics

Defining Asset Numbers, *Oracle Enterprise Asset Management User's Guide*

eAM Asset Genealogy Open Interface

The eAM Asset Genealogy Open Interface enables you to import asset genealogy (configuration history) into eAM, using a batch process. Create new parent/child relationships, or update existing relationships.

Execution Steps:

1. Populate the interface tables with the import information.

   The Asset Genealogy Import process reads information within the MTL_OBJECT_GENEALOGY_INTERFACE (MOGI) table, then imports that information into the production tables.

Column Name (partial list of columns)	Instruction
BATCH_ID	Enter an arbitrary number. Rows designated with the same BATCH_ID will process together.
PROCESS_STATUS	Enter P (Pending). This value will change to S if the import is successful, or E if the row contains an error.
INTERFACE_HEADER_ID	Unique key
IMPORT_MODE	Enter 0 to create new rows (configuration histories), or 1 to update existing rows.
OBJECT_TYPE	Enter 2 for eAM.
PARENT_OBJECT_TYPE	Enter 2 for eAM.
GENEALOGY_ORIGIN	Enter 3 (Manual) for eAM.
GENEALOGY_TYPE	Enter 5 for eAM.
INVENTORY_ITEM_ID	Corresponds to Asset Group
SERIAL_NUMBER	Corresponds to Asset Number
PARENT_INVENTORY_ITEM_ID	Corresponds to Parent Asset Group

Column Name (partial list of columns)	Instruction
PARENT_SERIAL_NUMBER	Corresponds to Parent Asset Number
START_DATE_ACTIVE	Enter the Parent/Child relationship start date.
END_DATE_ACTIVE	Enter the Parent/Child relationship end date.
ERROR_CODE	This column will update by the Import process if an error occurs.
ERROR_MESSAGE	This column will update by the Import process if an error occurs.

Note: For information about columns not discussed, see Table and View Definitions, *Oracle Enterprise Asset Management Technical Reference Manual.*

2. Launch the Asset Genealogy Import process.

    1. Navigate to the Asset Genealogy Import window.

        Choose an organization if you have not specified one already. Import asset genealogies into the master organization before importing them into additional children organizations.

    2. Enter parameters:

Parameter	Description
Batch ID	This is the same value that is populated in the BATCH_ID column within the MOGI table.

Parameter	Description
Purge Option	Select Yes to delete rows in the interface tables after they have successfully imported into the production tables. Select No to keep all rows in the interface tables after they have successfully imported into the production tables. Any failed rows with error messages will not delete.

3. Choose OK.

4. Choose Submit to launch the Asset Genealogy Import process. View its progress by choosing View from the tool bar, and then selecting Requests.

eAM Meter Reading Open Interface

Import Meter Reading is an interface process used to import meter readings into eAM.

Execution Steps:

1. Populate the interface table with the import information.

   The Meter Reading Import process reads information within the EAM_METER_READING_INTERFACE table, then imports that information into the eAM production table.

Column Name (partial list of columns)	Instruction
GROUP_ID	Enter an arbitrary number. Rows designated with the same GROUP_ID will process together.
PROCESS_STATUS	Enter P (Pending). This value will change to S if the import is successful, or E if the row contains an error.
PROCESS_PHASE	Enter 2 for rows to be processed.
METER_ID	Unique key

Column Name (partial list of columns)	Instruction
METER_NAME	Enter the name of the meter to process. This is not mandatory if you entered a METER_ID.
RESET_FLAG	Enter Yes to reset the meter reading. If there is an already existing meter reading for the given meter that occurs after this reset reading, the processor will error. Enter No to not reset the meter reading.
LIFE_TO_DATE	Enter a value for LIFE_TO_DATE or a value in the READING_VALUE column. If both columns are populated, then the READING_VALUE value is used to enter a reading, and the LIFE_TO_DATE_READING value is calculated from this current reading.
ORGANIZATION_ID	If you enter an organization ID, the organization provided must be enabled for eAM.
ORGANIZATION_CODE	If you enter an organization code, the organization provided must be enabled for eAM.
READING_DATE	Date the reading is entered
READING_VALUE	Reading value at the reading date
WIP_ENTITY_ID	Enter the work order id that the reading is associated with. If the column is populated, then the ORGANIZATION_ID or ORGANIZATION_CODE columns are mandatory.
WORK_ORDER_NAME	Enter the work order name that the reading is associated with. If this column is populated, then the ORGANIZATION_ID or ORGANIZATION_CODE columns are mandatory.

Column Name (partial list of columns)	Instruction
DESCRIPTION	Description of meter reading
ERROR_CODE	This column will update by the Import process if an error occurs.
ERROR_MESSAGE	This column will update by the Import process if an error occurs.

Note: For information about columns not discussed, see Table and View Definitions, *Oracle Enterprise Asset Management Technical Reference Manual.*

2. Launch the Meter Reading Import process.
    1. Navigate to the Import Jobs and Schedules window.

        Choose an organization if you have not specified one already. Import Work Orders into the master organization before importing them into additional children organizations.

    2. Enter parameters:

Parameter	Description
Group ID	Corresponds to the GROUP_ID in the EAM_METER_READING_INTERFACE table. The import process will only process those meter readings that have a GROUP_ID in the interface table matching the value entered in this parameter.

    3. Choose OK.

    4. Choose Submit to launch the Meter Reading Import process. View its progress by choosing View from the tool bar, and then selecting Requests.

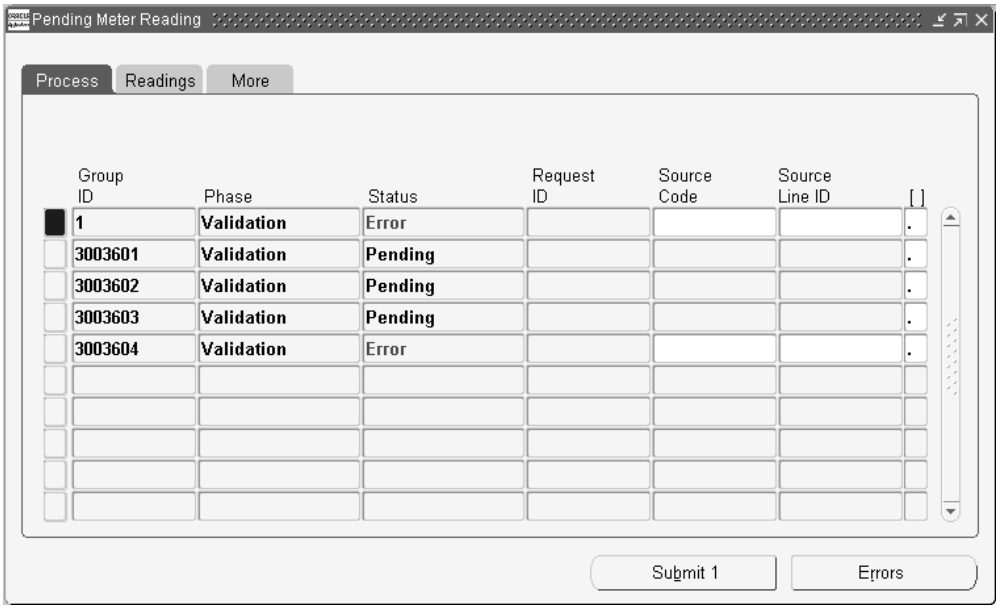
To view pending meter readings:

You can display the rows within the interface table that failed to import into eAM.

1. Navigate to the Find Meter Readings window to limit rows using search criteria, such as Group ID, Reading Date, or Organization code.

2. Choose Find. The rows that appear failed to import into eAM.

Pending Meter Reading



3. Optionally select the Process tab to display general information about the errored meter readings.

4. Optionally select the Readings tab to display meter reading information, such as meter name, reading date, and reading value.

5. Optionally select the More tab to display information about the meter reading, such as Organization, Work order, Description, and Created By.

6. Optionally select Errors to view additional detailed information regarding the type and cause of the failure.

7. Optionally choose Submit to import Work Orders, after correcting errors.

Related Topics

Entering Meter Readings, *Oracle Enterprise Asset Management User's Guide*

Asset Number API

Package Name:

EAM_AssetNumber_PUB

Procedure Name:

Insert_Asset_Number

The Insert_Asset_Number API is used to create Asset Numbers. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_inventory_item_id	NUMBER	Yes	-	Inventory Item ID
p_serial_number	VARCHAR2	Yes	-	Serial Number

Parameter	Type	Required	Default	Description
p_current_status	NUMBER	-	3	Current Status 3 - activated 4 - deactivated
p_descriptive_text	VARCHAR2	-	NULL	Description
p_current_organization_id	NUMBER	Yes	-	Current Organization ID
p_attribute_category	VARCHAR2	-	-	Descriptive Flexfield
p_attribute1	VARCHAR2	-	-	Descriptive Flexfield
p_attribute2	VARCHAR2	-	-	Descriptive Flexfield
p_attribute3	VARCHAR2	-	-	Descriptive Flexfield
p_attribute4	VARCHAR2	-	-	Descriptive Flexfield
p_attribute5	VARCHAR2	-	-	Descriptive Flexfield
p_attribute6	VARCHAR2	-	-	Descriptive Flexfield
p_attribute7	VARCHAR2	-	-	Descriptive Flexfield
p_attribute8	VARCHAR2	-	-	Descriptive Flexfield
p_attribute9	VARCHAR2	-	-	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute10	VARCHAR2	-	-	Descriptive Flexfield
p_attribute11	VARCHAR2	-	-	Descriptive Flexfield
p_attribute12	VARCHAR2	-	-	Descriptive Flexfield
p_attribute13	VARCHAR2	-	-	Descriptive Flexfield
p_attribute14	VARCHAR2	-	-	Descriptive Flexfield
p_attribute15	VARCHAR2	-	-	Descriptive Flexfield
p_wip_accounting_class_code	VARCHAR2	-	NULL	WIP Accounting Class Code
p_maintainable_flag	VARCHAR2	-	NULL	Maintainable Flag: Y, N, or Null
p_owning_department_id	NUMBER	Yes	-	Owning Department ID
p_network_asset_flag	VARCHAR2	-	NULL	Network Asset Flag: Y, N, or Null
p_fa_asset_id	NUMBER	-	NULL	Fixed Asset ID
p_pn_location_id	NUMBER	-	NULL	PN Location ID
p_eam_location_id	NUMBER	-	NULL	EAM Location ID
p_asset_criticality_code	VARCHAR2	-	NULL	Asset Criticality Code

Parameter	Type	Required	Default	Description
p_category_id	NUMBER	-	NULL	Category ID
p_prod_organization_id	NUMBER	-	NULL	Production Organization ID
p_equipment_item_id	NUMBER	-	NULL	Equipment Item ID
p_eqp_serial_number	VARCHAR2	-	NULL	Equipment Serial Number
p_instantiate_flag	BOOLEAN	-	FALSE	Instantiation Flag

Package Name:

EAM_AssetNumber_PUB

Procedure Name:

Update_Asset_Number

The EAM_AssetNumber_PUB.Update_Asset_Number API is used to update existing eAM Asset Numbers. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_inventory_item_id	NUMBER	Yes	-	Inventory Item ID
p_serial_number	VARCHAR2	Yes	-	Serial Number
p_current_status	NUMBER	-	3	Current Status 3 - activated 4 - deactivated
p_descriptive_text	VARCHAR2	-	NULL	Description
p_current_organization_id	NUMBER	Yes	-	Current Organization ID
p_attribute_category	VARCHAR2	-	-	Descriptive Flexfield
p_attribute1	VARCHAR2	-	-	Descriptive Flexfield
p_attribute2	VARCHAR2	-	-	Descriptive Flexfield
p_attribute3	VARCHAR2	-	-	Descriptive Flexfield
p_attribute4	VARCHAR2	-	-	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute5	VARCHAR2	-	-	Descriptive Flexfield
p_attribute6	VARCHAR2	-	-	Descriptive Flexfield
p_attribute7	VARCHAR2	-	-	Descriptive Flexfield
p_attribute8	VARCHAR2	-	-	Descriptive Flexfield
p_attribute9	VARCHAR2	-	-	Descriptive Flexfield
p_attribute10	VARCHAR2	-	-	Descriptive Flexfield
p_attribute11	VARCHAR2	-	-	Descriptive Flexfield
p_attribute12	VARCHAR2	-	-	Descriptive Flexfield
p_attribute13	VARCHAR2	-	-	Descriptive Flexfield
p_attribute14	VARCHAR2	-	-	Descriptive Flexfield
p_attribute15	VARCHAR2	-	-	Descriptive Flexfield
p_wip_accounting_class_code	VARCHAR2	-	NULL	WIP Accounting Class Code
p_maintainable_flag	VARCHAR2	-	NULL	Maintainable Flag: Y, N, or Null
p_owning_department_id	NUMBER	Yes	-	Owning Department ID

Parameter	Type	Required	Default	Description
p_network_asset_flag	VARCHAR2	-	NULL	Network Asset Flag: Y, N, or Null
p_fa_asset_id	NUMBER	-	NULL	Fixed Asset ID
p_pn_location_id	NUMBER	-	NULL	PN Location ID
p_eam_location_id	NUMBER	-	NULL	EAM Location ID
p_asset_criticality_code	VARCHAR2	-	NULL	Asset Criticality Code
p_category_id	NUMBER	-	NULL	Category ID
p_prod_organization_id	NUMBER	-	NULL	Production Organization ID
p_equipment_item_id	NUMBER	-	NULL	Equipment Item ID
p_eqp_serial_number	VARCHAR2	-	NULL	Equipment Serial Number

Asset Attribute Values API

Package Name:

EAM_ASSETATTR_VALUE_PUB

Procedure Name:

insert_assetattr_value

The EAM_ASSETATTR_VALUE_PUB.insert_assetattr_value API is used to create eAM Asset Attributes for existing Asset Numbers. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_association_id	NUMBER	Yes	-	Association ID
p_application_id	NUMBER	Yes	-	Application ID
p_descriptive_flexfield_name	VARCHAR2	Yes	-	Descriptive Flexfield Name
p_inventory_item_id	NUMBER	Yes	-	Inventory Item ID
p_serial_number	VARCHAR2	Yes	-	Serial Number
p_organization_id	NUMBER	Yes	-	Organization ID
p_attribute_category	VARCHAR2	Yes	-	Attribute Category

Parameter	Type	Required	Default	Description
p_c_attribute1	VARCHAR2	Yes	-	Attribute
p_c_attribute2	VARCHAR2	Yes	-	Attribute
p_c_attribute3	VARCHAR2	Yes	-	Attribute
p_c_attribute4	VARCHAR2	Yes	-	Attribute
p_c_attribute5	VARCHAR2	Yes	-	Attribute
p_c_attribute6	VARCHAR2	Yes	-	Attribute
p_c_attribute7	VARCHAR2	Yes	-	Attribute
p_c_attribute8	VARCHAR2	Yes	-	Attribute
p_c_attribute9	VARCHAR2	Yes	-	Attribute
p_c_attribute10	VARCHAR2	Yes	-	Attribute
p_c_attribute11	VARCHAR2	Yes	-	Attribute
p_c_attribute12	VARCHAR2	Yes	-	Attribute
p_c_attribute13	VARCHAR2	Yes	-	Attribute
p_c_attribute14	VARCHAR2	Yes	-	Attribute
p_c_attribute15	VARCHAR2	Yes	-	Attribute
p_c_attribute16	VARCHAR2	Yes	-	Attribute
p_c_attribute17	VARCHAR2	Yes	-	Attribute
p_c_attribute18	VARCHAR2	Yes	-	Attribute
p_c_attribute19	VARCHAR2	Yes	-	Attribute
p_c_attribute20	VARCHAR2	Yes	-	Attribute

Parameter	Type	Required	Default	Description
p_d_attribute1	DATE	Yes	-	Attribute
p_d_attribute2	DATE	Yes	-	Attribute
p_d_attribute3	DATE	Yes	-	Attribute
p_d_attribute4	DATE	Yes	-	Attribute
p_d_attribute5	DATE	Yes	-	Attribute
p_d_attribute6	DATE	Yes	-	Attribute
p_d_attribute7	DATE	Yes	-	Attribute
p_d_attribute8	DATE	Yes	-	Attribute
p_d_attribute9	DATE	Yes	-	Attribute
p_d_attribute10	DATE	Yes	-	Attribute
p_n_attribute1	NUMBER	Yes	-	Attribute
p_n_attribute2	NUMBER	Yes	-	Attribute
p_n_attribute3	NUMBER	Yes	-	Attribute
p_n_attribute4	NUMBER	Yes	-	Attribute
p_n_attribute5	NUMBER	Yes	-	Attribute
p_n_attribute6	NUMBER	Yes	-	Attribute
p_n_attribute7	NUMBER	Yes	-	Attribute
p_n_attribute8	NUMBER	Yes	-	Attribute
p_n_attribute9	NUMBER	Yes	-	Attribute
p_n_attribute10	NUMBER	Yes	-	Attribute

Parameter	Type	Required	Default	Description
p_maintenance_object_type	VARCHAR2	Yes	-	Maintenance Object Type
p_maintenance_object_id	NUMBER	Yes	-	Maintenance Object ID
p_creation_organization_id	NUMBER	Yes	-	Creation Organization ID

Package Name:

EAM_ASSETATTR_VALUE_PUB

Procedure Name:

update_assetattr_value

The EAM_ASSETATTR_VALUE_PUB.update_assetattr_value API is used to update eAM Asset Attributes for existing Asset Numbers. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter

Parameter	Type	Required	Default	Description
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_association_id	NUMBER	Yes	-	Association ID
p_application_id	NUMBER	Yes	-	Application ID
p_descriptive_flexfield_name	VARCHAR2	Yes	-	Descriptive Flexfield Name
p_inventory_item_id	NUMBER	Yes	-	Inventory Item ID
p_serial_number	VARCHAR2	Yes	-	Serial Number
p_organization_id	NUMBER	Yes	-	Organization ID
p_attribute_category	VARCHAR2	Yes	-	Attribute Category
p_c_attribute1	VARCHAR2	Yes	-	Attribute
p_c_attribute2	VARCHAR2	Yes	-	Attribute
p_c_attribute3	VARCHAR2	Yes	-	Attribute
p_c_attribute4	VARCHAR2	Yes	-	Attribute
p_c_attribute5	VARCHAR2	Yes	-	Attribute
p_c_attribute6	VARCHAR2	Yes	-	Attribute
p_c_attribute7	VARCHAR2	Yes	-	Attribute
p_c_attribute8	VARCHAR2	Yes	-	Attribute

Parameter	Type	Required	Default	Description
p_c_attribute9	VARCHAR2	Yes	-	Attribute
p_c_attribute10	VARCHAR2	Yes	-	Attribute
p_c_attribute11	VARCHAR2	Yes	-	Attribute
p_c_attribute12	VARCHAR2	Yes	-	Attribute
p_c_attribute13	VARCHAR2	Yes	-	Attribute
p_c_attribute14	VARCHAR2	Yes	-	Attribute
p_c_attribute15	VARCHAR2	Yes	-	Attribute
p_c_attribute16	VARCHAR2	Yes	-	Attribute
p_c_attribute17	VARCHAR2	Yes	-	Attribute
p_c_attribute18	VARCHAR2	Yes	-	Attribute
p_c_attribute19	VARCHAR2	Yes	-	Attribute
p_c_attribute20	VARCHAR2	Yes	-	Attribute
p_d_attribute1	DATE	Yes	-	Attribute
p_d_attribute2	DATE	Yes	-	Attribute
p_d_attribute3	DATE	Yes	-	Attribute
p_d_attribute4	DATE	Yes	-	Attribute
p_d_attribute5	DATE	Yes	-	Attribute
p_d_attribute6	DATE	Yes	-	Attribute
p_d_attribute7	DATE	Yes	-	Attribute
p_d_attribute8	DATE	Yes	-	Attribute

Parameter	Type	Required	Default	Description
p_d_attribute9	DATE	Yes	-	Attribute
p_d_attribute10	DATE	Yes	-	Attribute
p_n_attribute1	NUMBER	Yes	-	Attribute
p_n_attribute2	NUMBER	Yes	-	Attribute
p_n_attribute3	NUMBER	Yes	-	Attribute
p_n_attribute4	NUMBER	Yes	-	Attribute
p_n_attribute5	NUMBER	Yes	-	Attribute
p_n_attribute6	NUMBER	Yes	-	Attribute
p_n_attribute7	NUMBER	Yes	-	Attribute
p_n_attribute8	NUMBER	Yes	-	Attribute
p_n_attribute9	NUMBER	Yes	-	Attribute
p_n_attribute10	NUMBER	Yes	-	Attribute
p_maintenance_object_type	VARCHAR2	Yes	-	Maintenance Object Type
p_maintenance_object_id	NUMBER	Yes	-	Maintenance Object ID
p_creation_organization_id	NUMBER	Yes	-	Creation Organization ID

Asset Attribute Groups API

Package Name:

EAM_ASSETATTR_GRP_PUB

Procedure Name:

insert_assetattr_grp

The EAM_ASSETATTR_GRP_PUB.insert_assetattr_grp public API is used to associate existing eAM Asset Groups with existing Attribute Groups. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_application_id	NUMBER	Yes	-	Application ID
p_descriptive_flexfield_name	VARCHAR2	Yes	-	Descriptive Flexfield Name
p_desc_flex_context_code	VARCHAR2	Yes	-	Descriptive Flex Content Code
p_organization_id	NUMBER	Yes	-	Organization ID

Parameter	Type	Required	Default	Description
p_inventory_item_id	NUMBER	Yes	-	Inventory Item ID
p_enabled_flag	VARCHAR2	Yes	-	Enabled Flag
p_creation_organization_id	NUMBER	Yes	-	Creation Organization ID
x_new_association_id	NUMBER	-	-	The newly created Association ID

Package Name:

EAM_ASSETATTR_GRP_PUB

Procedure Name:

update_assetattr_grp

The EAM_ASSETATTR_GRP_PUB.update_assetattr_grp public API is used to update existing Asset Attribute Group associations. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_association_id	NUMBER	Yes	-	Association ID of the updated record
p_application_id	NUMBER	Yes	-	Application ID
p_descriptive_flexfield_name	VARCHAR2	Yes	-	Descriptive Flexfield Name
p_desc_flex_context_code	VARCHAR2	Yes	-	Descriptive Flex Content Code
p_organization_id	NUMBER	Yes	-	Organization ID
p_inventory_item_id	NUMBER	Yes	-	Inventory Item ID
p_enabled_flag	VARCHAR2	Yes	-	Enabled Flag
p_creation_organization_id	NUMBER	Yes	-	Creation Organization ID

Asset Routes API

Package Name:

EAM_ASSET_ROUTES_PUB

Procedure Name:

insert_asset_routes

The EAM_ASSET_ROUTES_PUB.insert_asset_routes public API is used to create new Asset Routes. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_organization_id_in	NUMBER	Yes	-	Organization ID
p_start_date_active_in	DATE	-	NULL	Start date active
p_end_date_active_in	DATE	-	NULL	End date active
p_attribute_category_in	VARCHAR2	-	NULL	Attribute Category

Parameter	Type	Required	Default	Description
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_network_item_id_in	NUMBER	Yes	-	Network Item ID
p_network_serial_number_in	VARCHAR2	Yes	-	Network Serial Number
p_inventory_item_id_in	NUMBER	Yes	-	Inventory Item ID
p_serial_number	VARCHAR2	Yes	-	Serial Number
p_network_object_type_in	NUMBER	-	NULL	Network Object Type
p_network_object_id_in	NUMBER	-	NULL	Network Object ID
p_maintenance_object_type	NUMBER	-	NULL	Maintenance Object Type
p_maintenance_object_id	NUMBER	-	NULL	Maintenance Object ID

Package Name:

EAM_ASSET_ROUTES_PUB

Procedure Name:

update_asset_routes

The EAM_ASSET_ROUTES_PUB.update_asset_routes public API is used to update existing Asset Routes. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_organization_id_in	NUMBER	Yes	-	Organization ID
p_start_date_active_in	DATE	-	NULL	Start date active
p_end_date_active_in	DATE	-	NULL	End date active
p_attribute_category_in	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_network_item_id_in	NUMBER	Yes	-	Network Item ID
p_network_serial_number_in	VARCHAR2	Yes	-	Network Serial Number

Parameter	Type	Required	Default	Description
p_inventory_item_id_in	NUMBER	Yes	-	Inventory Item ID
p_serial_number	VARCHAR2	Yes	-	Serial Number
p_network_object_type_in	NUMBER	-	NULL	Network Object Type
p_network_object_id_in	NUMBER	-	NULL	Network Object ID
p_maintenance_object_type	NUMBER	-	NULL	Maintenance Object Type
p_maintenance_object_id	NUMBER	-	NULL	Maintenance Object ID

Asset Areas API

Package Name:

EAM_ASSET_AREAS_PUB

Procedure Name:

insert_asset_areas

The EAM_ASSET_AREAS_PUB.insert_asset_areas public API is used to create new Asset Areas. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_location_codes	VARCHAR2	Yes	-	Location Code
p_start_date	DATE	-	NULL	Start Effective Date
p_end_date	DATE	-	NULL	End Effective Date
p_organization_id	NUMBER	Yes	-	Organization ID
p_description	VARCHAR2	-	NULL	Description
p_creation_organization_id	NUMBER	Yes	-	Creation Organization ID

Package Name:

EAM_ASSET_AREAS_PUB

Procedure Name:

update_asset_areas

The EAM_ASSET_AREAS_PUB.update_asset_areas public API is used to update existing Asset Areas. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	-	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_location_codes	VARCHAR2	Yes	-	Location Code
p_start_date	DATE	-	NULL	Start Effective Date

Parameter	Type	Required	Default	Description
p_end_date	DATE	-	NULL	End Effective Date
p_organization_id	NUMBER	Yes	-	Organization ID
p_description	VARCHAR2	-	NULL	Description
p_creation_organization_id	NUMBER	Yes	-	Creation Organization ID

Department Approvers API

Package Name:

EAM_DEPT_APPROVERS_PUB

Procedure Name:

insert_dept_appr

The EAM_DEPT_APPROVERS_PUB.insert_dept_appr public API is used to create new department approvers. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_dept_id	NUMBER	Yes	-	Department ID
p_organization_id	NUMBER	Yes	-	Organization ID
p_resp_app_id	NUMBER	Yes	-	Application ID
p_responsibility_id	NUMBER	Yes	-	Responsibility ID
p_primary_approver_id	NUMBER	Yes	-	Primary Approver ID

Package Name:

EAM_DEPT_APPROVERS_PUB

Procedure Name:

update_dept_appr

The EAM_DEPT_APPROVERS_PUB.update_dept_appr public API is used to update existing department approvers. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_dept_id	NUMBER	Yes	-	Department ID
p_organization_id	NUMBER	Yes	-	Organization ID
p_resp_app_id	NUMBER	Yes	-	Application ID
p_responsibility_id	NUMBER	Yes	-	Responsibility ID
p_primary_approver_id	NUMBER	Yes	-	Primary Approver ID

EAM Parameters API

Package Name:

EAM_PARAMETERS_PUB

Procedure Name:

Insert_Parameters

The EAM_PARAMETERS_PUB.Insert_Parameters public API is used to create a new set of eAM Parameters for an existing eAM enabled organization. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_organization_id	NUMBER	Yes	-	Organization ID
p_work_request_auto_approve	VARCHAR2	-	'N'	Work Request Auto Approve
p-def_maint_cost_category	NUMBER	Yes	-	Maintenance Cost Category

Parameter	Type	Required	Default	Description
p_def_eam_cost_element_id	NUMBER	Yes	-	eAM Cost Element ID
p_work_req_extended_log_flag	VARCHAR2	-	'Y'	Work Request Extended Log flag
p_default_eam_class	VARCHAR2	Yes	-	WIP Accounting Class
p_easy_work_order_prefix	VARCHAR2	-	NULL	Easy Work Order prefix
p_work_order_prefix	VARCHAR2	-	NULL	Work Order prefix
p_serial_number_enabled	VARCHAR2	-	'Y'	Serial Number enabled
p_auto_firm_flag	VARCHAR2	-	'Y'	Auto Firm flag
p_maintenance_offset_account	NUMBER	-	NULL	Maintenance Offset Account
p_wip_eam_request_type	NUMBER	-	NULL	WIP eAM Request Type
p_material_issue_by_mo	VARCHAR2	-	'N'	Material Issue flag
p_default_department_id	NUMBER	-	NULL	Default Department ID
p_invoice_billable_items_only	VARCHAR2	-	'N'	Invoice Billable Items Only
p_override_bill_amount	VARCHAR2	-	NULL	Override Bill Amount
p_billing_basis	NUMBER	-	NULL	Billing Basis

Parameter	Type	Required	Default	Description
p_billing_method	NUMBER	-	NULL	Billing Method
p_dynamic_billing_activity	VARCHAR2	-	NULL	Dynamic Billing Activity
p_default_asset_flag	VARCHAR2	-	NULL	Default Asset flag
p_pm_ignore_missed_wo	VARCHAR2	-	'N'	flag indicating if PM scheduling should ignore missed Work Orders
p_issue_zero_cost_flag	VARCHAR2	-	'Y'	flag indicating if you are issuing rebuldables at zero or item cost
p_work_request_asset_num_reqd	VARCHAR2	-	'Y'	flag indicating if an Asset Number is mandatory at Work Request creation

Package Name:

EAM_PARAMETERS_PUB

Procedure Name:

Update_Parameters

The EAM_PARAMETERS_PUB.Update_Parameters public API is used to update an existing set of eAM Parameters. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_organization_id	NUMBER	Yes	-	Organization ID
p_work_request_auto_approve	VARCHAR2	-	'N'	Work Request Auto Approve
p-def_maint_cost_category	NUMBER	Yes	-	Maintenance Cost Category
p_def_eam_cost_element_id	NUMBER	Yes	-	eAM Cost Element ID
p_work_req_extended_log_flag	VARCHAR2	-	'Y'	Work Request Extended Log flag
p_default_eam_class	VARCHAR2	Yes	-	WIP Accounting Class

Parameter	Type	Required	Default	Description
p_easy_work_order_prefix	VARCHAR2	-	NULL	Easy Work Order prefix
p_work_order_prefix	VARCHAR2	-	NULL	Work Order prefix
p_serial_number_enabled	VARCHAR2	-	'Y'	Serial Number enabled
p_auto_firm_flag	VARCHAR2	-	'Y'	Auto Firm flag
p_maintenance_offset_account	NUMBER	-	NULL	Maintenance Offset Account
p_wip_eam_request_type	NUMBER	-	NULL	WIP eAM Request Type
p_material_issue_by_mo	VARCHAR2	-	'N'	Material Issue flag
p_default_department_id	NUMBER	-	NULL	Default Department ID
p_invoice_billable_items_only	VARCHAR2	-	'N'	Invoice Billable Items Only
p_override_bill_amount	VARCHAR2	-	NULL	Override Bill Amount
p_billing_basis	NUMBER	-	NULL	Billing Basis
p_billing_method	NUMBER	-	NULL	Billing Method
p_dynamic_billing_activity	VARCHAR2	-	NULL	Dynamic Billing Activity
p_default_asset_flag	VARCHAR2	-	NULL	Default Asset flag

Parameter	Type	Required	Default	Description
p_pm_ignore_missed_wo	VARCHAR2	-	'N'	flag indicating if PM scheduling should ignore missed Work Orders
p_issue_zero_cost_flag	VARCHAR2	-	'Y'	flag indicating if you are issuing rebuldables at zero or item cost
p_work_request_asset_num_reqd	VARCHAR2	-	'Y'	flag indicating if an Asset Number is mandatory at Work Request creation

EAM Meters API

Package Name:

EAM_METER_PUB

Procedure Name:

create_meter

The EAM_METER_PUB.create_meter public API is used to create new meters. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_meter_name	VARCHAR2	Yes	-	Meter Name
p_meter_uom	VARCHAR2	Yes	-	Unit of Measure
p_meter_type	NUMBER	-	1	Meter Type 1 = value meter 2 = change meter
p_value_change_dir	NUMBER	-	1	Value change direction 1 = ascending 2 = descending
p_used_in_scheduling	NUMBER	-	'N'	Used in Scheduling flag
p_user_defined_rate	NUMBER	-	NULL	User Defined Rate
p_use_past_reading	NUMBER	-	NULL	Use Past Reading
p_description	VARCHAR2	-	NULL	Meter Description

Parameter	Type	Required	Default	Description
p_from_effective_date	DATE	-	NULL	From Effective Date
p_to_effective_date	DATE	-	NULL	To Effective Date
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_templ_flag	VARCHAR2	-	NULL	Template flag
p_source_tmpl_id	NUMBER	-	NULL	Source Template ID
p_initial_reading	NUMBER	-	0	Initial Reading
p_initial_reading_date	DATE	-	SYSDATE	Initial Reading Date
x_new_meter_id	NUMBER	-	-	Newly created Meter's ID

Package Name:

EAM_METER_PUB

Procedure Name:

update_meter

The EAM_METER_PUB.update_meter public API is used to update existing meters. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_meter_name	VARCHAR2	Yes	-	Meter Name
p_meter_uom	VARCHAR2	Yes	-	Unit of Measure
p_meter_type	NUMBER	-	1	Meter Type 1 = value meter 2 = change meter
p_value_change_dir	NUMBER	-	1	Value change direction 1 = ascending 2 = descending
p_used_in_scheduling	NUMBER	-	'N'	Used in Scheduling flag
p_user_defined_rate	NUMBER	-	NULL	User Defined Rate

Parameter	Type	Required	Default	Description
p_use_past_reading	NUMBER	-	NULL	Use Past Reading
p_description	VARCHAR2	-	NULL	Meter Description
p_from_effective_date	DATE	-	NULL	From Effective Date
p_to_effective_date	DATE	-	NULL	To Effective Date
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_templ_flag	VARCHAR2	-	NULL	Template flag
p_source_tmpl_id	NUMBER	-	NULL	Source Template ID

EAM Meter Association API

Package Name:

EAM_MeterAssoc_PUB

Procedure Name:

Insert_AssetMeterAssoc

The EAM_MeterAssoc_PUB.Insert_AssetMeterAssoc public API is used to create new meter associations. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_meter_id	NUMBER	Yes	-	Meter ID
p_organization_id	NUMBER	Yes	-	Organization ID
p_asset_group_id	NUMBER	-	NULL	Inventory Item ID
p_asset_number	VARCHAR2	-	NULL	Serial Number
p_maintenance_object_type	NUMBER	-	NULL	Maintenance Object Type
p_maintenance_object_id	NUMBER	-	NULL	Maintenance Object ID

Note: Because all fields (other than WHO columns and attribute columns) within the eam_asset_meters table are part of the unique key, no update method is supplied.

Meter Reading API

Package Name:

EAM_MeterReading_PUB

Procedure Name:

create_meter_reading

The create_meter_reading API is used to create new meter readings and reset existing meter readings. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
p_meter_reading_rec	Eam_MeterReading_PUB.meter_reading_Rec_Type	Yes	-	The record includes details of the meter reading.

Parameter	Type	Required	Default	Description
p_value_before_reset	NUMBER	-	NULL	Value of the meter reading before reset. Required only when the reading is a reset.
x_meter_reading_id	NUMBER	-	-	The meter_reading_id of the newly created record.

Column Name	Type	Default
meter_id	NUMBER	NULL
meter_reading_id	NUMBER	NULL
current_reading	NUMBER	NULL
current_reading_date	DATE	NULL
reset_flag	VARCHAR2(1)	NULL
description	VARCHAR2(100)	NULL
wip_entity_id	NUMBER	NULL
attribute_category	VARCHAR2(30)	NULL
attribute1	VARCHAR2(150)	NULL
attribute2	VARCHAR2(150)	NULL
attribute3	VARCHAR2(150)	NULL
attribute4	VARCHAR2(150)	NULL

Column Name	Type	Default
attribute5	VARCHAR2(150)	NULL
attribute6	VARCHAR2(150)	NULL
attribute7	VARCHAR2(150)	NULL
attribute8	VARCHAR2(150)	NULL
attribute9	VARCHAR2(150)	NULL
attribute10	VARCHAR2(150)	NULL
attribute11	VARCHAR2(150)	NULL
attribute12	VARCHAR2(150)	NULL
attribute13	VARCHAR2(150)	NULL
attribute14	VARCHAR2(150)	NULL
attribute15	VARCHAR2(150)	NULL
source_line_id	NUMBER	NULL
source_code	VARCHAR2(30)	NULL
wo_entry_fake_flag	VARCHAR2(1)	NULL

Package Name:

EAM_MeterReading_PUB

Procedure Name:

disable_meter_reading

The disable_meter_reading API is used to disable existing meter readings. You need to supply either a meter reading ID or a meter reading date, to identify the specific reading. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
P_API_VERSION	NUMBER	Yes	-	Standard Oracle API parameter
P_INIT_MSG_LIST	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_COMMIT	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_VALIDATION_LEVEL	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
X_RETURN_STATUS	VARCHAR2	-	-	Standard Oracle API output parameter
X_MSG_COUNT	NUMBER	-	-	Standard Oracle API output parameter
X_MSG_DATA	VARCHAR2	-	-	Standard Oracle API output parameter
p_meter_reading_id	NUMBER	-	NULL	The meter_reading_id of the meter reading to be disabled.
p_meter_id	NUMBER	-	NULL	The meter_id of the meter.
p_meter_reading_date	DATE	-	NULL	Meter reading date
p_meter_name	VARCHAR2	-	NULL	Meter name

EAM PM Schedules API (includes PM Rules as children records)

Package Name:

EAM_PMDef_PUB

Procedure Name:

instantiate_PM_def

Given an Activity association instance's Activity association ID and a PM Template's pm_schedule_id, the EAM_PMDef_PUB.instantiate_PM_def public API is used to create a new PM definition from the template, where the new PM definition is associated with the given activity_association_id. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_pm_schedule_id	NUMBER	Yes	-	the PM Template's pm_schedule_id
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
p_activity_assoc_id	NUMBER	Yes	-	the Activity association instance's Activity association ID
x_new_pm_schedule_id	NUMBER	-	-	the newly created PM Schedule's pm_schedule_id

Package Name:

EAM_PMDef_PUB

Procedure Name:

instantiate_PM_Defs

The EAM_PMDef_PUB.instantiate_PM_Defs public API is used to instantiate a set of

PM definitions for all asset_association_ids within the activity_assoc_id_tbl table and create a PM Schedule definition from the template. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_activity_assoc_id_tbl	EAM_ObjectInstantiation_PUB.Association_Id_Tbl_Type	-	-	PL/SQL table type containing one or more association_Id_Tbl_Type rows

Package Name

EAM_PMDef_PUB

Procedure Name:

create_PM_Def

The EAM_PMDef_PUB.create_PM_Def public API is used to create new PM Schedules and child records, such as PM rules. The table below provides the specifications for this

API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_pm_schedule_rec	pm_scheduling_rec_type	Yes	-	PL/SQL table type containing one unique PM Schedule record
p_pm_day_interval_rules_tbl	pm_rule_tbl_type	Yes	-	PL/SQL table type containing zero or more PL/SQL record type pm_rule_rec_type rows, used for creating Day Interval type runtime rules

Parameter	Type	Required	Default	Description
p_pm_runtime_rules_tbl	pm_rule_tbl_type	Yes	-	PL/SQL table type containing zero or more PL/SQL record type pm_rule_rec_type rows, used for creating meter-based runtime rules
p_pm_list_date_rules_tbl	pm_rule_tbl_type	Yes	-	PL/SQL table type containing zero or more PL/SQL record type pm_rule_rec_type rows, used to create simple list date rules.
x_new_pm_schedule_id	NUMBER	-	-	pm_schedule_id of the newly created PM Schedule

Package Name:

EAM_PMDef_PUB

Procedure Name:

update_PM_Def

The EAM_PMDef_PUB.update_PM_Def public API is used to update existing PM Schedules and child records, such as PM rules. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_pm_schedule_rec	pm_scheduling_rec_type	-	NULL	PL/SQL table type containing one unique PM Schedule record. Unlike the create procedure, this may be NULL because this procedure can be used to modify PM Rules.
p_pm_day_interval_rules_tbl	pm_rule_tbl_type	Yes	-	PL/SQL table type containing zero or more PL/SQL record type pm_rule_rec_type rows, used for creating Day Interval type runtime rules

Parameter	Type	Required	Default	Description
p_pm_runtime_rules_tbl	pm_rule_tbl_type	Yes	-	PL/SQL table type containing zero or more PL/SQL record type pm_rule_rec_type rows, used for creating meter-based runtime rules
p_pm_list_date_rules_tbl	pm_rule_tbl_type	Yes	-	PL/SQL table type containing zero or more PL/SQL record type pm_rule_rec_type rows, used to create simple list date rules.

Package Name:

EAM_PMDef_PUB

Procedure Name:

PM_Scheduling_Rec_Type

The EAM_PMDef_PUB.PM_Scheduling_Rec_Type is a PL SQL record type used for inserting PM Schedule definitions into the eam_pm_schedulings table.

Column	Type
PM_SCHEDULE_ID	NUMBER
ACTIVITY_ASSOCIATION_ID	NUMBER
NON_SCHEDULED_FLAG	VARCHAR2(1)
FROM_EFFECTIVE_DATE	DATE

Column	Type
TO_EFFECTIVE_DATE	DATE
RESCHEDULING_POINT	NUMBER
LEAD_TIME	NUMBER
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
DAY_TOLERANCE	NUMBER

Column	Type
SOURCE_CODE	VARCHAR2(30)
SOURCE_LINE	VARCHAR2(30)
DEFAULT_IMPLEMENT	VARCHAR2(1)
WHICHEVER_FIRST	VARCHAR2(1)
INCLUDE_MANUAL	VARCHAR2(1)
SET_NAME_ID	NUMBER
SCHEDULING_METHOD_CODE	NUMBER
TYPE_CODE	NUMBER
NEXT_SERVICE_START_DATE	DATE
NEXT_SERVICE_END_DATE	DATE
SOURCE_TMPL_ID	NUMBER
AUTO_INSTANTIATION_FLAG	VARCHAR2(1)
NAME	VARCHAR (50)
TMPL_FLAG	VARCHAR2(1)

Package Name:

EAM_PMDef_PUB

Procedure Name:

pm_rule_rec_type

The EAM_PMDef_PUB.pm_rule_rec_type is a PL SQL record type used for inserting PM Schedule rules that are associated with a PM Schedule into the eam_pm_scheduling_rules table.

Column	Type
RULE_ID	NUMBER
PM_SCHEDULE_ID	NUMBER
RULE_TYPE	NUMBER
DAY_INTERVAL	NUMBER
METER_ID	NUMBER
RUNTIME_INTERVAL	NUMBER
LAST_SERVICE_READING	NUMBER
EFFECTIVE_READING_FROM	NUMBER
EFFECTIVE_READING_TO	NUMBER
EFFECTIVE_DATE_FROM	DATE
EFFECTIVE_DATE_TO	DATE
LIST_DATE	DATE
LIST_DATE_DESC	VARCHAR2(50)

Activity Creation API

Package Name:

EAM_ACTIVITY_PUB

Procedure Name:

CREATE_ACTIVITY

The Activity Creation API is used to create eAM Activities. You can specify the source Work Order that the API uses as a model for the new Activities. Optionally provide an Item Template to define the attributes of the Activity. You can specify various Activity properties, such as Activity Type, Cause, Shutdown Notification, and Source. Various copy options, controlling the copy of Operations, Material, Resources, and Activity

Association, are supported.

> **Note:** Items (Asset Groups, Activities, Rebuildables) are created using the Item Creation Business Object API (See: Item Creation Business Object API, *Oracle Manufacturing APIs and Open Interfaces Manual*). Asset BOMs and Asset Activities are created using the BOM Business Object API (See: BOM Business Object API, *Oracle Manufacturing APIs and Open Interfaces Manual*).

The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
P_API_VERSION	NUMBER	Yes	-	Standard Oracle API parameter
P_INIT_MSG_LIST	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_COMMIT	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_VALIDATION_LEVEL	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
X_RETURN_STATUS	VARCHAR2	-	-	Standard Oracle API output parameter
X_MSG_COUNT	NUMBER	-	-	Standard Oracle API output parameter
X_MSG_DATA	VARCHAR2	-	-	Standard Oracle API output parameter
P_ASSET_ACTIVITY	INV_ITEM_GRP.ITEM_REC_TYPE	Yes	-	Item Record to define the attributes of the Activity

Parameter	Type	Required	Default	Description
P_TEMPLATE_ID	NUMBER	-	NULL	Template Id (If Template Name is specified, Template Id will override Template Name)
P_TEMPLATE_NAME	VARCHAR2	-	NULL	Template Name
P_ACTIVITY_TYPE_CODE	VARCHAR2	-	NULL	Activity Type
P_ACTIVITY_CAUSE_CODE	VARCHAR2	-	NULL	Activity Cause
P_SHUTDOWN_TYPE_CODE	VARCHAR2	-	NULL	Shutdown Type
P_NOTIFICATION_REQ_FLAG	VARCHAR2	-	NULL	Notification Required Flag: Y (enabled), N (disabled)
_ACTIVITY_SOURCE_CODE	VARCHAR2	-	NULL	Activity Source
P_WORK_ORDER_REC	EAM_ACTIVITY_PUB.WORK_ORDER_REC_TYPE	Yes	-	Specifies the source Work Order the new Activity is to be created from
P_OPERATION_COPY_OPTION	NUMBER	-	2	Operation Copy Option: 1 (NONE), 2 (ALL)
P_MATERIAL_COPY_OPTION	NUMBER	-	2	Material Copy Option: 1 (NONE), 2 (ISSUED), 3 (ALL)

Parameter	Type	Required	Default	Description
P_RESOURCE_COPY_OPTION	NUMBER	-	2	Resource Copy Option: 1 (NONE), 2 (ISSUED), 3 (ALL)
P_ASSOCIATION_COPY_OPTION	NUMBER	-	2	Association Copy Option: 1 (NONE), 2 (CURRENT), 3 (ALL)
X_WORK_ORDER_REC	EAM_ACTIVITY_PUB.WORK_ORDER_REC_TYPE	-	-	Output. Validated Work Order record.
X_CURR_ITEM_REC	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Validated current item record.
X_CURR_ITEM_RETURN_STATUS	VARCHAR2	-	-	Output. Current item creation return status.
X_CURR_ITEM_ERROR_TBL	INV_ITEM_GRP.ERROR_TBL_TYPE	-	-	Output. Current item creation error table.
X_MASTER_ITEM_REC	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Validated master item record.
X_MASTER_ITEM_RETURN_STATUS	VARCHAR2	-	-	Output. Master item creation return status.
X_MASTER_ITEM_ERROR_TBL	INV_ITEM_GRP.ERROR_TBL_TYPE	-	-	Output. Master item creation error table.

Parameter	Type	Required	Default	Description
x_rtg_header_rec	BOM_Rtg_Pub.Rtg_Header_Rec_Type	-	-	Routing Business Object API output.
x_rtg_revision_tbl	BOM_Rtg_Pub.Rtg_Revision_Tbl_Type	-	-	Routing Business Object API output.
x_operation_tbl	BOM_Rtg_Pub.Operation_Tbl_Type	-	-	Routing Business Object API output.
x_op_resource_tbl	BOM_Rtg_Pub.Op_Resource_Tbl_Type	-	-	Routing Business Object API output.
x_sub_resource_tbl	BOM_Rtg_Pub.Sub_Resource_Tbl_Type	-	-	Routing Business Object API output.
x_op_network_tbl	BOM_Rtg_Pub.Op_Network_Tbl_Type	-	-	Routing Business Object API output.
x_rtg_return_status	VARCHAR2	-	-	Routing Business Object API output.
x_rtg_msg_count	NUMBER	-	-	Routing Business Object API output.
x_rtg_msg_list	Error_Handler.Error_Tbl_Type	-	-	Routing Business Object API output.
x_bom_header_rec	BOM_BO_PUB.BOM_Head_Rec_Type	-	-	BOM Business Object API output.
x_bom_revision_tbl	BOM_BO_PUB.BOM_Revision_Tbl_Type	-	-	BOM Business Object API output.

Parameter	Type	Required	Default	Description
x_bom_component_tbl	BOM_BO_PUB.BOM_Co_mps_Tbl_Type	-	-	BOM Business Object API output.
x_bom_return_status	VARCHAR2	-	-	BOM Business Object API output.
x_bom_msg_count	NUMBER	-	-	BOM Business Object API output.
x_bom_msg_list	Error_Handler.Error_Tbl_Type	-	-	BOM Business Object API output.
x_assoc_return_status	VARCHAR2	-	-	Copy Association API output.
x_assoc_msg_count	NUMBER	-	-	Copy Association API output.
x_assoc_msg_data	VARCHAR2	-	-	Copy Association API output.
x_act_num_association_tbl	EAM_Activity_PUB.Activity_Association_Tbl_Type	-	-	Copy Association API output.
x_activity_association_tbl	EAM_Activity_PUB.Activity_Association_Tbl_Type	-	-	Copy Association API output.

Package Name:

EAM_ACTIVITY_PUB

Procedure Name:

COPY_ACTIVITY

The Copy Activity API is used to create a new Activity from an existing Activity. While copying from the source Activity, you can copy the source Activity's BOM, Routing, and associations. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
P_API_VERSION	NUMBER	Yes	-	Standard Oracle API parameter
P_INIT_MSG_LIST	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_COMMIT	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_VALIDATION_LEVEL	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
X_RETURN_STATUS	VARCHAR2	-	-	Standard Oracle API output parameter
X_MSG_COUNT	NUMBER	-	-	Standard Oracle API output parameter
X_MSG_DATA	VARCHAR2	-	-	Standard Oracle API output parameter
P_ASSET_ACTIVITY	INV_ITEM_GRP.ITEM_REC_TYPE	Yes	-	Item Record to define the attributes of the Activity
P_TEMPLATE_ID	NUMBER	-	NULL	Template Id (If Template Name is specified, Template Id will override Template Name)

Parameter	Type	Required	Default	Description
P_TEMPLATE_NAME	VARCHAR2	-	NULL	Template Name
P_ACTIVITY_TYPE_CODE	VARCHAR2	-	NULL	Activity Type
P_ACTIVITY_CAUSE_CODE	VARCHAR2	-	NULL	Activity Cause
P_SHUTDOWN_TYPE_CODE	VARCHAR2	-	NULL	Shutdown Type
P_NOTIFICATION_REQ_FLAG	VARCHAR2	-	NULL	Notification Required Flag: Y (enabled), N (disabled)
_ACTIVITY_SOURCE_CODE	VARCHAR2	-	NULL	Activity Source
P_WORK_ORDER_REC	EAM_ACTIVITY_PUB.WORK_ORDER_REC_TYPE	Yes	-	Specifies the source Work Order the new Activity is to be created from
P_OPERATION_COPY_OPTION	NUMBER	-	2	Operation Copy Option: 1 (NONE), 2 (ALL)
P_MATERIAL_COPY_OPTION	NUMBER	-	2	Material Copy Option: 1 (NONE), 2 (ISSUED), 3 (ALL)
P_RESOURCE_COPY_OPTION	NUMBER	-	2	Resource Copy Option: 1 (NONE), 2 (ISSUED), 3 (ALL)

Parameter	Type	Required	Default	Description
P_ASSOCIATION_COPY_OPTION	NUMBER	-	2	Association Copy Option: 1 (NONE), 2 (CURRENT), 3 (ALL)
X_WORK_ORDER_REC	EAM_ACTIVITY_PUB.WORK_ORDER_REC_TYPE	-	-	Output. Validated Work Order record.
X_CURR_ITEM_REC	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Validated current item record.
X_CURR_ITEM_RETURN_STATUS	VARCHAR2	-	-	Output. Current item creation return status.
X_CURR_ITEM_ERROR_TBL	INV_ITEM_GRP.ERROR_TBL_TYPE	-	-	Output. Current item creation error table.
X_MASTER_ITEM_REC	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Validated master item record.
X_MASTER_ITEM_RETURN_STATUS	VARCHAR2	-	-	Output. Master item creation return status.
X_MASTER_ITEM_ERROR_TBL	INV_ITEM_GRP.ERROR_TBL_TYPE	-	-	Output. Master item creation error table.
x_rtg_header_rec	BOM_Rtg_Pub.Rtg_Header_Rec_Type	-	-	Routing Business Object API output.
x_rtg_revision_tbl	BOM_Rtg_Pub.Rtg_Revision_Tbl_Type	-	-	Routing Business Object API output.

Parameter	Type	Required	Default	Description
x_operation_tbl	BOM_Rtg_Pub.Operation_Tbl_Type	-	-	Routing Business Object API output.
x_op_resource_tbl	BOM_Rtg_Pub.Op_Resource_Tbl_Type	-	-	Routing Business Object API output.
x_sub_resource_tbl	BOM_Rtg_Pub.Sub_Resource_Tbl_Type	-	-	Routing Business Object API output.
x_op_network_tbl	BOM_Rtg_Pub.Op_Network_Tbl_Type	-	-	Routing Business Object API output.
x_rtg_return_status	VARCHAR2	-	-	Routing Business Object API output.
x_rtg_msg_count	NUMBER	-	-	Routing Business Object API output.
x_rtg_msg_list	Error_Handler.Error_Tbl_Type	-	-	Routing Business Object API output.
x_bom_header_rec	BOM_BO_PUB.BOM_Head_Rec_Type	-	-	BOM Business Object API output.
x_bom_revision_tbl	BOM_BO_PUB.BOM_Revision_Tbl_Type	-	-	BOM Business Object API output.
x_bom_component_tbl	BOM_BO_PUB.BOM_Co_mps_Tbl_Type	-	-	BOM Business Object API output.
x_bom_return_status	VARCHAR2	-	-	BOM Business Object API output.

Parameter	Type	Required	Default	Description
x_bom_msg_count	NUMBER	-	-	BOM Business Object API output.
x_bom_msg_list	Error_Handler.Error_Tbl_Type	-	-	BOM Business Object API output.
x_assoc_return_status	VARCHAR2	-	-	Copy Association API output.
x_assoc_msg_count	NUMBER	-	-	Copy Association API output.
x_assoc_msg_data	VARCHAR2	-	-	Copy Association API output.
x_act_num_association_tbl	EAM_Activity_PUB.Activity_Association_Tbl_Type	-	-	Copy Association API output.
x_activity_association_tbl	EAM_Activity_PUB.Activity_Association_Tbl_Type	-	-	Copy Association API output.

Package Name:

EAM_ACTIVITY_PUB

Procedure Name:

CREATE_ACTIVITY_WITH_TEMPLATE

The Copy Activity API is used to create a new Activity from an pre-defined template. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
P_API_VERSION	NUMBER	Yes	-	Standard Oracle API parameter
P_INIT_MSG_LIST	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_COMMIT	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_VALIDATION_LEVEL	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
X_RETURN_STATUS	VARCHAR2	-	-	Standard Oracle API output parameter
X_MSG_COUNT	NUMBER	-	-	Standard Oracle API output parameter
X_MSG_DATA	VARCHAR2	-	-	Standard Oracle API output parameter
P_ORGANIZATION_ID	NUMBER	-	-	Organization id
P_ORGANIZATION_CODE	NUMBER	-	-	Organization code
P_ASSET_ACTIVITY	VARCHAR2	-	NULL	Activity
P_SEGMENT1	VARCHAR2	-	NULL	Segment1
P_SEGMENT2	VARCHAR2	-	NULL	Segment2
P_SEGMENT3	VARCHAR2	-	NULL	Segment3
P_SEGMENT4	VARCHAR2	-	NULL	Segment4

Parameter	Type	Required	Default	Description
P_SEGMENT5	VARCHAR2	-	NULL	Segment5
P_SEGMENT6	VARCHAR2	-	NULL	Segment6
P_SEGMENT7	VARCHAR2	-	NULL	Segment7
P_SEGMENT8	VARCHAR2	-	NULL	Segment8
P_SEGMENT9	VARCHAR2	-	NULL	Segment9
P_SEGMENT10	VARCHAR2	-	NULL	Segment10
P_SEGMENT11	VARCHAR2	-	NULL	Segment11
P_SEGMENT12	VARCHAR2	-	NULL	Segment12
P_SEGMENT13	VARCHAR2	-	NULL	Segment13
P_SEGMENT14	VARCHAR2	-	NULL	Segment14
P_SEGMENT15	VARCHAR2	-	NULL	Segment15
P_SEGMENT16	VARCHAR2	-	NULL	Segment16
P_SEGMENT17	VARCHAR2	-	NULL	Segment17
P_SEGMENT18	VARCHAR2	-	NULL	Segment18
P_SEGMENT19	VARCHAR2	-	NULL	Segment19
P_SEGMENT20	VARCHAR2	-	NULL	Segment20
P_DESCRIPTION	VARCHAR2	-	-	Description
P_TEMPLATE_ID	NUMBER	-	NULL	Template id
P_TEMPLATE_NAME	VARCHAR2	-	NULL	Name of template

Parameter	Type	Required	Default	Description
P_ACTIVITY_TYPE_CODE	VARCHAR2	-	NULL	Activity code
P_ACTIVITY_CAUSE_CODE	VARCHAR2	-	NULL	Activity cause
P_SHUTDOWN_TYPE_CODE	VARCHAR2	-	NULL	Shutdown Type
P_NOTIFICATION_REQ_FLAG	VARCHAR2	-	NULL	Notification Required Flag (Y for enabled, N for disabled)
P_ACTIVITY_SOURCE_CODE	VARCHAR2	-	NULL	Activity Source
X_CURR_ITEM_REC	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Validated current item record.
X_CURR_ITEM_RETURN_STATUS	VARCHAR2	-	-	Output. Current item creation return status.
X_CURR_ITEM_ERROR_TBL	INV_ITEM_GRP.ITEM_ERROR_TBL_TYPE	-	-	Output. Current item creation error table.
X_MASTER_ITEM_REC	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Validated master item record.
X_MASTER_ITEM_RETURN_STATUS	VARCHAR2	-	-	Output. Master item creation return status.
X_MASTER_ITEM_ERROR_TBL	INV_ITEM_GRP.ITEM_REC_TYPE	-	-	Output. Master item creation error table.

Column Name	Type	Default
organization_id	NUMBER	NULL
organization_code	VARCHAR2(3)	NULL
wip_entity_id	NUMBER	NULL
wip_entity_name	VARCHAR2(240)	NULL

Column Name	Type	Default
organization_id	NUMBER	NULL
asset_activity_id	NUMBER	NULL
start_date_active	DATE	NULL
end_date_active	DATE	NULL
priority_code	VARCHAR2(30)	NULL
attribute_category	VARCHAR2(30)	NULL
attribute1	VARCHAR2(150)	NULL
attribute2	VARCHAR2(150)	NULL
attribute3	VARCHAR2(150)	NULL
attribute4	VARCHAR2(150)	NULL
attribute5	VARCHAR2(150)	NULL
attribute6	VARCHAR2(150)	NULL
attribute7	VARCHAR2(150)	NULL
attribute8	VARCHAR2(150)	NULL

Column Name	Type	Default
attribute9	VARCHAR2(150)	NULL
attribute10	VARCHAR2(150)	NULL
attribute11	VARCHAR2(150)	NULL
attribute12	VARCHAR2(150)	NULL
attribute13	VARCHAR2(150)	NULL
attribute14	VARCHAR2(150)	NULL
attribute15	VARCHAR2(150)	NULL
owning_department_id	NUMBER	NULL
activity_cause_code	VARCHAR2(30)	NULL
activity_type_code	VARCHAR2(30)	NULL
activity_source_code	VARCHAR2(30)	NULL
class_code	VARCHAR2(10)	NULL
maintenance_object_id	NUMBER	NULL
genealogy_id	NUMBER	NULL
inventory_item_id	NUMBER	NULL
serial_number	VARCHAR2(30)	NULL
activity_association_id	NUMBER	NULL
tagging_required_flag	VARCHAR2(1)	NULL
shutdown_type_code	VARCHAR2(30)	NULL
templ_flag	VARCHAR2(1)	NULL

Column Name	Type	Default
creation_organization_id	NUMBER	NULL
return_status	VARCHAR2(1)	NULL
error_mesg	VARCHAR2(240)	NULL

TYPE activity_association_tbl_type IS TABLE OF activity_association_rec_type

INDEX BY BINARY_INTEGER

Related Topics

BOM Business Object API*, Oracle Manufacturing APIs and Open Interfaces Manual*

Item Creation Business Object API*, Oracle Manufacturing APIs and Open Interfaces Manual*

EAM Activity Association API

Package Name:

EAM_ITEM_ACTIVITIES_PUB

Procedure Name:

Insert_item_activities

The EAM_ITEM_ACTIVITIES_PUB.Insert_item_activities public API is used to create new Activity associations. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_asset_activity_id	NUMBER	Yes	-	Asset Activity ID
p_inventory_item_id	NUMBER	-	NULL	Inventory Item ID
p_organization_id	NUMBER	Yes	-	Organization ID
p_owningdepartment_id	NUMBER	-	NULL	Owning Department ID
p_maintenance_object_id	NUMBER	-	NULL	Maintenance Object ID
p_creation_organization_id	NUMBER	-	NULL	Creation Organization ID
p_start_date_active	DATE	-	NULL	Start Date Active
p_end_date_active	DATE	-	NULL	End Date Active
p_priority_code	VARCHAR2	-	NULL	Priority Code

Parameter	Type	Required	Default	Description
p_activity_cause_code	VARCHAR2	-	NULL	Activity Cause code
p_activity_type_code	VARCHAR2	-	NULL	Activity Type code
p_shutdown_type_code	VARCHAR2	-	NULL	Shutdown Type code
p_maintenance_object_type	NUMBER	-	NULL	Maintenance Object Type
p_tmpl_flag	VARCHAR2	-	NULL	Template flag
p_class_code	VARCHAR2	-	NULL	Class code
p_activity_source_code	VARCHAR2	-	NULL	Activity Source code
p_serial_number	VARCHAR2	-	NULL	Serial Number
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_tagging_required_flag	VARCHAR2	-	NULL	Tagging Required flag
p_last_service_start_date	DATE	-	NULL	Last Service Start Date
p_last_service_end_date	DATE	-	NULL	Last Service End Date
p_prev_service_start_date	DATE	-	NULL	Previous Service Start Date
p_prev_service_end_date	DATE	-	NULL	Previous Service End Date

Parameter	Type	Required	Default	Description
p_source_tmpl_id	NUMBER	-	NULL	Source Template ID
p_pm_last_service_tbl	EAM_PM_LAST_SERVICE_PUB.pm_last_service_tbl	Yes	-	PM Last Service table

Package Name:

EAM_ITEM_ACTIVITIES_PUB

Procedure Name:

Update_item_activities

The EAM_ITEM_ACTIVITIES_PUB.Update_item_activities public API is used to update existing Activity associations. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_asset_activity_id	NUMBER	Yes	-	Asset Activity ID
p_inventory_item_id	NUMBER	-	NULL	Inventory Item ID
p_organization_id	NUMBER	Yes	-	Organization ID
p_owningdepartment_id	NUMBER	-	NULL	Owning Department ID
p_maintenance_object_id	NUMBER	-	NULL	Maintenance Object ID
p_creation_organization_id	NUMBER	-	NULL	Creation Organization ID
p_start_date_active	DATE	-	NULL	Start Date Active
p_end_date_active	DATE	-	NULL	End Date Active
p_priority_code	VARCHAR2	-	NULL	Priority Code
p_activity_cause_code	VARCHAR2	-	NULL	Activity Cause code
p_activity_type_code	VARCHAR2	-	NULL	Activity Type code
p_shutdown_type_code	VARCHAR2	-	NULL	Shutdown Type code
p_maintenance_object_type	NUMBER	-	NULL	Maintenance Object Type

Parameter	Type	Required	Default	Description
p_tmpl_flag	VARCHAR2	-	NULL	Template flag
p_class_code	VARCHAR2	-	NULL	Class code
p_activity_source_code	VARCHAR2	-	NULL	Activity Source code
p_serial_number	VARCHAR2	-	NULL	Serial Number
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_tagging_required_flag	VARCHAR2	-	NULL	Tagging Required flag
p_last_service_start_date	DATE	-	NULL	Last Service Start Date
p_last_service_end_date	DATE	-	NULL	Last Service End Date
p_prev_service_start_date	DATE	-	NULL	Previous Service Start Date
p_prev_service_end_date	DATE	-	NULL	Previous Service End Date
p_source_tmpl_id	NUMBER	-	NULL	Source Template ID
p_pm_last_service_tbl	EAM_PM_LAST_SERVICE_PUB.pm_last_service_tbl	Yes	-	PM Last Service table

EAM Activity Suppression API

Package Name:

EAM_ActivitySupn_PUB

Procedure Name:

Insert_ActivitySupn

The EAM_ActivitySupn_PUB.Insert_ActivitySupn public API is used to create new Activity Suppressions. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_parent_association_id	NUMBER	Yes	-	Parent Association ID
p_child_association_id	NUMBER	Yes	-	Child Association ID

Parameter	Type	Required	Default	Description
p_tmpl_flag	VARCHAR2	-	NULL	Template flag
p_description	VARCHAR2	-	NULL	Description
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield

Package Name:

EAM_ActivitySupn_PUB

Procedure Name:

Update_ActivitySupn

The EAM_ActivitySupn_PUB.Update_ActivitySupn public API is used to update existing Activity Suppressions. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_parent_association_id	NUMBER	Yes	-	Parent Association ID
p_child_association_id	NUMBER	Yes	-	Child Association ID
p_tmpl_flag	VARCHAR2	-	NULL	Template flag
p_description	VARCHAR2	-	NULL	Description
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield

EAM Set Name API

Package Name:

EAM_SetName_PUB

Procedure Name:

Insert_PMSetName

The EAM_SetName_PUB.Insert_PMSetName public API is used to create new Set Names. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter

Parameter	Type	Required	Default	Description
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_set_name	VARCHAR2	Yes	-	Set Name
p_description	VARCHAR2	-	NULL	Description
p_end_date	DATE	-	NULL	End Date
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
x_new_set_name	NUMBER	-	-	Newly create Set Name's ID

Parameter	Type	Required	Default	Description
p_end_date_val_req	VARCHAR2	-	'true'	Flag indicating if the validation end date is in the future. This flag is 'true' by default, and is 'false' only when the API is called by the iSetup API.

Package Name:

EAM_SetName_PUB

Procedure Name:

Update_PMSetName

The EAM_SetName_PUB.Update_PMSetName public API is used to update existing Set Names. The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter

Parameter	Type	Required	Default	Description
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_set_name_id	NUMBER	Yes	-	Set Name of the record to update
p_set_name	VARCHAR2	Yes	-	Set Name
p_description	VARCHAR2	-	NULL	Description
p_end_date	DATE	-	NULL	End Date
p_attribute_category	VARCHAR2	-	NULL	Attribute Category
p_attribute1	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute2	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute3	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute4	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute5	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute6	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute7	VARCHAR2	-	NULL	Descriptive Flexfield

Parameter	Type	Required	Default	Description
p_attribute8	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute9	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute10	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute11	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute12	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute13	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute14	VARCHAR2	-	NULL	Descriptive Flexfield
p_attribute15	VARCHAR2	-	NULL	Descriptive Flexfield
p_end_date_val_req	VARCHAR2	-	'true'	Flag indicating if the validation end date is in the future. This flag is 'true' by default, and is 'false' only when the API is called by the iSetup API.

Maintenance Object Instantiation API

Package Name:

EAM_OBJECTINSTANTIATION_PUB

Procedure Name:

INSTANTIATE_OBJECT

The Maintenance Object Instantiation API is first triggered after the creation of a Maintenance Object. It will then call the private packages for the Activity Instantiation (from the Maintenance Item/Activity Templates), and Preventive Maintenance and Meter Instantiations (from their templates). The table below provides the specifications for this API:

Parameter	Type	Required	Default	Description
P_API_VERSION	NUMBER	Yes	-	Standard Oracle API parameter
P_INIT_MSG_LIST	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_COMMIT	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
P_VALIDATION_LEVEL	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
X_RETURN_STATUS	VARCHAR2	-	-	Standard Oracle API output parameter
X_MSG_COUNT	NUMBER	-	-	Standard Oracle API output parameter
X_MSG_DATA	VARCHAR2	-	-	Standard Oracle API output parameter
P_MAINTENANCE_OBJECT_ID	NUMBER	Yes	-	Standard Oracle API output parameter
P_MAINTENANCE_OBJECT_TYPE	NUMBER	Yes	-	Supports Type 1 (Serial Numbers)

Parameter	Type	Required	Default	Description
P_COMMIT	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter

TYPE association_id_tbl_type IS TABLE OF number

INDEX BY BINARY_INTEGER

Work Order Business Object API

Package Name:

EAM_PROCESS_WO_PUB

Procedure Name:

Process_Master_Child_WO

The EAM_PROCESS_WO_PUB.Process_Master_Child_WO public API is used to create and update Work Orders, Work Order Relationships, Operations, Operation Networks, Material, Resources, and Resource instances. The table below provides the specifications of this API:

Parameter	Type	Required	Default	Description
p_po_identifier	VARCHAR2	Yes	EAM	Standard Oracle API parameter
p_api_version_number	NUMBER	Yes	1.0	Standard Oracle API parameter
p_init_msg_list	BOOLEAN	Yes	FALSE	Standard Oracle API parameter
p_eam_wo_relations_tbl	EAM_PROCESS_WO_PUB.eam_wo_relations_tbl_type	Yes	-	PL SQL table containing one or more records that identify the relationship between Work Orders

Parameter	Type	Required	Default	Description
p_eam_wo_tbl	EAM_PROCESS_WO_PUB.eam_wo_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Orders to process
p_eam_op_tbl	EAM_PROCESS_WO_PUB.eam_op_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order operations to process
p_eam_op_network_tbl	EAM_PROCESS_WO_PUB.eam_op_network_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order operation networks to process
p_eam_res_tbl	EAM_PROCESS_WO_PUB.eam_res_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order resource requirements to process
p_eam_res_inst_tbl	EAM_PROCESS_WO_PUB.eam_res_inst_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order resource instances to process

Parameter	Type	Required	Default	Description
p_eam_sub_res_tbl	EAM_PROCESS_WO_PUB.eam_sub_res_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order substitute resources to process
p_eam_mat_req_tbl	EAM_PROCESS_WO_PUB.eam_mat_req_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order material requirements to process
p_eam_direct_items_tbl	EAM_PROCESS_WO_PUB.eam_direct_items_tbl_type	Yes	-	PL/SQL table containing one or more records that identify Work Order description-based direct items to process
x_eam_wo_tbl	EAM_PROCESS_WO_PUB.eam_wo_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Orders that were processed and their post-processing attributes

Parameter	Type	Required	Default	Description
x_eam_wo_relations_tbl	EAM_PROCESS_WO_PUB.eam_wo_relations_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order relationships that were processed and their post-processing attributes
x_eam_op_tbl	EAM_PROCESS_WO_PUB.eam_op_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order operations that were processed and their post-processing attributes
x_eam_op_network_tbl	EAM_PROCESS_WO_PUB.eam_op_network_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order operation networks that were processed and their post-processing attributes

Parameter	Type	Required	Default	Description
x_eam_res_tbl	EAM_PROCESS_WO_PUB.eam_res_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order resource requirements that were processed and their post-processing attributes
x_eam_res_inst_tbl	EAM_PROCESS_WO_PUB.eam_res_inst_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order resource instances that were processed and their post-processing attributes
x_eam_sub_res_tbl	EAM_PROCESS_WO_PUB.eam_sub_res_tbl	Yes	-	output PL/SQL table containing one or more records that identify Work Order substitute resources that were processed and their post-processing attributes

Parameter	Type	Required	Default	Description
x_eam_mat_req_tbl	EAM_PROCESS_WO_PUB.eam_mat_req_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order material requirements that were processed and their post-processing attributes
x_eam_direct_items_tbl	EAM_PROCESS_WO_PUB.eam_direct_items_tbl_type	Yes	-	output PL/SQL table containing one or more records that identify Work Order description-based direct items that were processed and their post-processing attributes
x_return_status	VARCHAR2	Yes	-	Standard Oracle API output parameter
x_msg_count	NUMBER	Yes	-	Standard Oracle API output parameter
p_commit	VARCHAR2	Yes	N	Standard Oracle API output parameter
p_debug	VARCHAR2	Yes	N	Standard Oracle API output parameter

Parameter	Type	Required	Default	Description
p_output_dir	VARCHAR2	Yes	NULL	The output directory where the debug messages are written to, if the debug parameter is set to Yes. This directory should be accessible by the database.
p_debug_filename	VARCHAR2	Yes	EAM_WO_DEBUG.log	The output file name where the debug messages are written to, if the debug parameter is set to Yes. The directory should be accessible by the database.
p_debug_file_mode	VARCHAR2	Yes	w	The debug file should open in this mode, for example, in append or overwrite mode.

Following are the columns that must pass for p_eam_wo_relations_tbl record type:

Column	Type
BATCH_ID	NUMBER
WO_RELATIONSHIP_ID	NUMBER
PARENT_OBJECT_ID	NUMBER
PARENT OBJECT_TYPE_ID	NUMBER
PARENT_HEADER_ID	NUMBER

Column	Type
CHILD_OBJECT_ID	NUMBER
CHILD_OBJECT_TYPE_ID	NUMBER
CHILD_HEADER_ID	NUMBER
PARENT_RELATIONSHIP_TYPE	NUMBER
RELATIONSHIP_STATUS	NUMBER
TOP_LEVEL_OBJECT_ID	NUMBER
TOP_LEVEL_OBJECT_TYPE_ID	NUMBER
TOP_LEVEL_HEADER_ID	NUMBER
ADJUST_PARENT	VARCHAR2(1)
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER
ROW_ID	NUMBER

Following are the columns that must pass for p_eam_wo_tbl record type:

Column	Type
HEADER_ID	NUMBER
BATCH_ID	NUMBER
ROW_ID	NUMBER
WIP_ENTITY_NAME	VARCHAR2(240)
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER

Column	Type
DESCRIPTION	VARCHAR2(240)
ASSET_NUMBER	VARCHAR2(30)
ASSET_GROUP_ID	NUMBER
REBUILD_ITEM_ID	NUMBER
REBUILD_SERIAL_NUMBER	VARCHAR2(30)
MAINTENANCE_OBJECT_ID	NUMBER
MAINTENANCE_OBJECT_TYPE	NUMBER
MAINTENANCE_OBJECT_SOURCE	NUMBER
CLASS_CODE	VARCHAR2(10)
ASSET_ACTIVITY_ID	NUMBER
ACTIVITY_TYPE	VARCHAR2(30)
ACTIVITY_CAUSE	VARCHAR2(30)
ACTIVITY_SOURCE	VARCHAR2(30)
WORK_ORDER_TYPE	VARCHAR2(30)
STATUS_TYPE	NUMBER
JOB_QUANTITY	NUMBER
DATE_RELEASED	DATE
OWNING_DEPARTMENT	NUMBER
PRIORITY	NUMBER
REQUESTED_START_DATE	DATE

Column	Type
DUE_DATE	DATE
SHUTDOWN_TYPE	VARCHAR2(30)
FIRM_PLANNED_FLAG	NUMBER
NOTIFICATION_REQUIRED	VARCHAR2(1)
TAGOUT_REQUIRED	VARCHAR2(1)
PLAN_MAINTENANCE	VARCHAR2(1)
PROJECT_ID	NUMBER
TASK_ID	NUMBER
END_ITEM_UNIT_NUMBER	VARCHAR2(30)
SCHEDULE_GROUP_ID	NUMBER
BOM_REVISION_DATE	DATE
ROUTING_REVISION_DATE	DATE
ALTERNATE_ROUTING_DESIGNATOR	VARCHAR2(10)
ALTERNATE_BOM_DESIGNATOR	VARCHAR2(10)
ROUTING_REVISION	VARCHAR2(3)
BOM_REVISION	VARCHAR2(3)
PARENT_WIP_ENTITY_ID	NUMBER
MANUAL_REBUILD_FLAG	VARCHAR2(1)
PM_SCHEDULE_ID	NUMBER
WIP_SUPPLY_TYPE	NUMBER

Column	Type
MATERIAL_ACCOUNT	NUMBER
MATERIAL_OVERHEAD_ACCOUNT	NUMBER
RESOURCE_ACCOUNT	NUMBER
OUTSIDE_PROCESSING_ACCOUNT	NUMBER
MATERIAL_VARIANCE_ACCOUNT	NUMBER
RESOURCE_VARIANCE_ACCOUNT	NUMBER
OUTSIDE_PROC_VARIANCE_ACCOUNT	NUMBER
STD_COST_ADJUSTMENT_ACCOUNT	NUMBER
OVERHEAD_ACCOUNT	NUMBER
OVERHEAD_VARIANCE_ACCOUNT	NUMBER
SCHEDULED_START_DATE	DATE
SCHEDULED_COMPLETION_DATE	DATE
COMMON_BOM_SEQUENCE_ID	NUMBER
COMMON_ROUTING_SEQUENCE_ID	NUMBER
PO_CREATION_TIME	NUMBER
GEN_OBJECT_ID	NUMBER
MATERIAL_ISSUE_BY_MO	VARCHAR2(1)
USER_ID	NUMBER
RESPONSIBILITY_ID	NUMBER
SOURCE_LINE_ID	NUMBER

Column	Type
SOURCE_CODE	VARCHAR2(30)
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
ISSUE_ZERO_COST_FLAG	VARCHAR2(1)
USER_ID	NUMBER
RESPONSIBILITY_ID	NUMBER

Column	Type
REQUEST_ID	NUMBER
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER
SOURCE_LINE_ID	NUMBER
SOURCE_CODE	VARCHAR2(30)
VALIDATE_STRUCTURE	VARCHAR2(1)
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_op_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER
OPERATION_SEQ_NUM	NUMBER
STANDARD_OPERATION_ID	NUMBER
DEPARTMENT_ID	NUMBER
OPERATION_SEQUENCE_ID	NUMBER
DESCRIPTION	VARCHAR2(240)

Column	Type
MINIMUM_TRANSFER_QUANTITY	NUMBER
COUNT_POINT_TYPE	NUMBER
BACKFLUSH_FLAG	NUMBER
SHUTDOWN_TYPE	VARCHAR2(30)
START_DATE	DATE
COMPLETION_DATE	DATE
LONG_DESCRIPTION	VARCHAR2(4000)
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)

Column	Type
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
REQUEST_ID	NUMBER
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_op_network_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER
PRIOR_OPERATION	NUMBER
NEXT_OPERATION	NUMBER
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)

Column	Type
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_mat_req_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER

Column	Type
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER
OPERATION_SEQ_NUM	NUMBER
INVENTORY_ITEM_ID	NUMBER
QUANTITY_PER_ASSEMBLY	NUMBER
DEPARTMENT_ID	NUMBER
WIP_SUPPLY_TYPE	NUMBER
DATE_REQUIRED	DATE
REQUIRED_QUANTITY	NUMBER
REQUESTED_QUANTITY	NUMBER
RELEASED_QUANTITY	NUMBER
QUANTITY_ISSUED	NUMBER
SUPPLY_SUBINVENTORY	VARCHAR2(10)
SUPPLY_LOCATOR_ID	NUMBER
MRP_NET_FLAG	NUMBER
MPS_REQUIRED_QUANTITY	NUMBER
MPS_DATE_REQUIRED	DATE
COMPONENT_SEQUENCE_ID	NUMBER
COMMENTS	VARCHAR2(240)
AUTO_REQUEST_MATERIAL	VARCHAR2(1)

Column	Type
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
SUGGESTED_VENDOR_NAME	VARCHAR2(240)
VENDOR_ID	NUMBER
UNIT_PRICE	NUMBER
REQUEST_ID	NUMBER

Column	Type
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER
PROGRAM_UPDATE_DATE	DATE
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_res_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER
OPERATION_SEQ_NUM	NUMBER
RESOURCE_SEQ_NUM	NUMBER
RESOURCE_ID	NUMBER
UOM_CODE	VARCHAR2(3)
BASIS_TYPE	NUMBER
USAGE_RATE_OR_AMOUNT	NUMBER
ACTIVITY_ID	NUMBER
SCHEDULED_FLAG	NUMBER

Column	Type
ASSIGNED_UNITS	NUMBER
AUTOCHARGE_TYPE	NUMBER
STANDARD_RATE_FLAG	NUMBER
APPLIED_RESOURCE_UNITS	NUMBER
APPLIED_RESOURCE_VALUE	NUMBER
START_DATE	DATE
COMPLETION_DATE	DATE
SCHEDULE_SEQ_NUM	NUMBER
SUBSTITUTE_GROUP_NUM	NUMBER
REPLACEMENT_GROUP_NUM	NUMBER
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)

Column	Type
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
DEPARTMENT_ID	NUMBER
REQUEST_ID	NUMBER
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER
PROGRAM_UPDATE_DATE	DATE
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_res_inst_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER

Column	Type
OPERATION_SEQ_NUM	NUMBER
RESOURCE_SEQ_NUM	NUMBER
INSTANCE_ID	NUMBER
SERIAL_NUMBER	VARCHAR2(30)
START_DATE	DATE
COMPLETION_DATE	DATE
TOP_LEVEL_BATCH_ID	NUMBER

Following are the columns that must pass for p_eam_sub_res_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER
OPERATION_SEQ_NUM	NUMBER
RESOURCE_SEQ_NUM	NUMBER
RESOURCE_ID	NUMBER
UOM_CODE	VARCHAR2(3)
BASIS_TYPE	NUMBER
USAGE_RATE_OR_AMOUNT	NUMBER

Column	Type
ACTIVITY_ID	NUMBER
SCHEDULED_FLAG	NUMBER
ASSIGNED_UNITS	NUMBER
AUTOCHARGE_TYPE	NUMBER
STANDARD_RATE_FLAG	NUMBER
APPLIED_RESOURCE_UNITS	NUMBER
APPLIED_RESOURCE_VALUE	NUMBER
START_DATE	DATE
COMPLETION_DATE	DATE
SCHEDULE_SEQ_NUM	NUMBER
SUBSTITUTE_GROUP_NUM	NUMBER
REPLACEMENT_GROUP_NUM	NUMBER
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)

Column	Type
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
DEPARTMENT_ID	NUMBER
REQUEST_ID	NUMBER
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER
PROGRAM_UPDATE_DATE	DATE
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_res_usage_tbl record type:

Column	Type
HEADER_ID	NUMBER
ROW_ID	NUMBER
BATCH_ID	NUMBER

Column	Type
WIP_ENTITY_ID	NUMBER
OPERATION_SEQ_NUM	NUMBER
RESOURCE_SEQ_NUM	NUMBER
ORGANIZATION_ID	NUMBER
START_DATE	DATE
COMPLETION_DATE	DATE
ASSIGNED_UNITS	NUMBER
INSTANCE_ID	NUMBER
SERIAL_NUMBER	VARCHAR2(30)
REQUEST_ID	NUMBER
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER
PROGRAM_UPDATE_DATE	DATE
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Following are the columns that must pass for p_eam_direct_items_tbl record type:

Column	Type
HEADER_ID	NUMBER
BATCH_ID	NUMBER
ROW_ID	NUMBER

Column	Type
DESCRIPTION	VARCHAR2(240)
PURCHASING_CATEGORY_ID	NUMBER
DIRECT_ITEM_SEQUENCE_ID	NUMBER
OPERATION_SEQ_NUM	NUMBER
DEPARTMENT_ID	NUMBER
WIP_ENTITY_ID	NUMBER
ORGANIZATION_ID	NUMBER
SUGGESTED_VENDOR_NAME	VARCHAR2(240)
SUGGESTED_VENDOR_ID	NUMBER
SUGGESTED_VENDOR_SITE	VARCHAR2(15)
SUGGESTED_VENDOR_SITE_ID	NUMBER
SUGGESTED_VENDOR_CONTACT	VARCHAR2(80)
SUGGESTED_VENDOR_CONTACT_ID	NUMBER
SUGGESTED_VENDOR_PHONE	VARCHAR2(20)
SUGGESTED_VENDOR_ITEM_NUM	VARCHAR2(25)
UNIT_PRICE	NUMBER
AUTO_REQUEST_MATERIAL	VARCHAR2(1)
REQUIRED_QUANTITY	NUMBER
REQUESTED_QUANTITY	NUMBER
UOM	VARCHAR2(3)

Column	Type
NEED_BY_DATE	DATE
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
REQUEST_ID	NUMBER
PROGRAM_ID	NUMBER
PROGRAM_APPLICATION_ID	NUMBER

Column	Type
PROGRAM_UPDATE_DATE	DATE
RETURN_STATUS	VARCHAR2(1)
TRANSACTION_TYPE	NUMBER

Work Request API

Package Name:

WIP_EAM_WORKREQUEST_PUB

Procedure Name:

Work_Request_Import

The WIP_EAM_WORKREQUEST_PUB.Work_Request_Import public API is used to create and update Work Requests. The table below provides the specifications of this API:

Parameter	Type	Required	Default	Description
p_api_version	NUMBER	Yes	-	Standard Oracle API parameter
p_init_msg_list	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_commit	VARCHAR2	-	FND_API.G_FALSE	Standard Oracle API parameter
p_validation_level	NUMBER		FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_return_status	VARCHAR2	-	-	Standard Oracle API output parameter

Parameter	Type	Required	Default	Description
x_msg_count	NUMBER	-	FND_API.G_VALID_LEVEL_FULL	Standard Oracle API parameter
x_msg_data	VARCHAR2	-	-	Standard Oracle API output parameter
p_mode	VARCHAR2	Yes	-	Import mode is 'CREATE' or 'UPDATE'
p_work_request_rec	WIP_EAM_WORKREQUESTS%ROWTYPE	Yes	-	Rowtype record from WIP_EAM_WORK_REQUESTS table
p_request_log	VARCHAR2	Yes	-	Work Request problem description that appends to the Work Request log, specified during 'CREATE' and 'UPDATE'
p_user_id	NUMBER	Yes	-	user ID of person creating Work Request
x_work_request_id	NUMBER	-	-	If import mode = 'CREATE', the Work Request ID for the newly created Work Request returns. If import mode = 'UPDATE', the Work Request ID of the Work Request being updated returns.

The following columns need to pass for p_work_request_rec:

Column	Type
WORK_REQUEST_ID	NUMBER
WORK_REQUEST_NUMBER	VARCHAR2(64)
ASSET_NUMBER	VARCHAR2(30)
ORGANIZATION ID	NUMBER
WORK_REQUEST_STATUS_ID	NUMBER
WORK_REQUEST_PRIORITY_ID	NUMBER
WORK_REQUEST_OWNING_DEPT	NUMBER
EXPECTED_RESOLUTION_DATE	DATE
ATTRIBUTE_CATEGORY	VARCHAR2(30)
ATTRIBUTE1	VARCHAR2(150)
ATTRIBUTE2	VARCHAR2(150)
ATTRIBUTE3	VARCHAR2(150)
ATTRIBUTE4	VARCHAR2(150)
ATTRIBUTE5	VARCHAR2(150)
ATTRIBUTE6	VARCHAR2(150)
ATTRIBUTE7	VARCHAR2(150)
ATTRIBUTE8	VARCHAR2(150)
ATTRIBUTE9	VARCHAR2(150)
ATTRIBUTE10	VARCHAR2(150)
ATTRIBUTE11	VARCHAR2(150)

Column	Type
ATTRIBUTE12	VARCHAR2(150)
ATTRIBUTE13	VARCHAR2(150)
ATTRIBUTE14	VARCHAR2(150)
ATTRIBUTE15	VARCHAR2(150)
WORK_REQUEST_TYPE_ID	NUMBER
WORK_REQUEST_CREATED_FOR	NUMBER
PHONE_NUMBER	VARCHAR2(4000)
E_MAIL	VARCHAR2(240)
CONTACT_PREFERENCE	NUMBER
NOTIFY_ORIGINATOR	NUMBER

18

eAM Profile Options

This chapter covers the following topics:

- Profile Option Summary
- Profile Option Details
- Profile Options in Other Applications
- Profile Option Details

Profile Option Summary

The table below indicates if you can view or update the profile option and which System Administrator levels the profile options can be updated: Site, Application, Responsibility, and User. Use the Personal Profile Options window to view or set your profile options at the user level. View the *Oracle Applications System Administrator's Guide* for a list of profile options common to all Oracle applications.

A Required profile option requires a value. An Optional profile option provides a default value; provide a value only if you want to change this value.

Profile Option Summary

Profile Option	User	System Admin: USER	System Admin: RESP	System Admin: APP	System Admin: SITE	Required ?	Default
EAM: Asset Description	View Update	View Update	View Update	View Update	View Update	N	No

Profile Option	User	System Admin: USER	System Admin: RESP	System Admin: APP	System Admin: SITE	Required ?	Default
EAM: Debug PM Scheduling	View Update	View Update	View Update	View Update	View Update	N	Yes
EAM: PM Debug File (including complete file path)	View Update	View Update	View Update	View Update	View Update	N	<Path determined during Rapid Install Setups>/eampmeng.log
EAM: Activity API Debug File Directory	View Update	View Update	View Update	View Update	View Update	N	-
EAM: Activity API Debug File Name	View Update	View Update	View Update	View Update	View Update	N	EAM_ABO_debug.log
EAM: Activity API Debug Option	View Update	View Update	View Update	View Update	View Update	N	No
EAM: Debug Profile Option	View Update	View Update	View Update	View Update	View Update	N	-

Profile Option	User	System Admin: USER	System Admin: RESP	System Admin: APP	System Admin: SITE	Required ?	Default
EAM: Object Instantiation API Log File Directory	View Update	View Update	View Update	View Update	View Update	N	-
EAM: Object Instantiation API Log File Name	View Update	View Update	View Update	View Update	View Update	N	EAM_MOI.log
EAM: Object Instantiation API Log Option	View Update	View Update	View Update	View Update	View Update	N	No
EAM: Maintenance Supervisor	View Update	View Update	View Update	View Update	View Update	N	No
EAM: Maintenance Work Request Options	View	View	View Update	View	View	N	-

Profile Option Details

eAM's profile options are listed below:

EAM: Activity API Debug File Directory

The Activity API is called in eAM when a new Activity is created from either a Work Order or Activity. This profile option indicates the debug file directory that the debug

log will reside, if debugging is enabled for the Activity API. For example, If the profile option, EAM: Activity API Debug Option, is set to Yes, the log of the activity process is saved in the directory determined by the path designated in the current profile option. This profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: Activity API Debug File Name

The Activity API is called in eAM when a new Activity is created from either a Work Order or Activity. This profile option indicates the debug file name that resides in the debug file directory, if debugging is enabled for the Activity API. For example, If the profile option, EAM: Activity API Debug Option, is set to Yes, the log of the activity process is saved in the directory determined by the path designated by the EAM: Activity API Debug File Directory profile option. The name of the file that includes the activity process log is determined by the current profile option. This profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: Activity API Debug Option

The Activity API is called in eAM when a new Activity is created from either a Work Order or Activity. This profile option indicates whether debugging is enabled for the Activity API. This profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: Asset Description

Use this profile option to determine whether you want the asset hierarchy view to display the asset's description, and the length of its description. By default, the description does not appear.

EAM: Debug PM Scheduling

This profile option indicates whether debug mode is enabled for Preventive Maintenance Scheduling processes. The profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels. The debug mode enables a tracing feature and causes additional messages to be printed to a log file. The default value is Yes. The internal name of the profile option is EAM_DEBUG_PM_SCHED. Available values are as follows:

- Yes - System is operating in debug mode
- No - System is operating in normal mode
- Blank (no value) - Equivalent to No

EAM: Debug Profile Option

This profile option is used by multiple eAM processes. If this profile is enabled, the log of these processes is collected. These processes include the following:

- Asset Number Import Program - the log is visible in the Concurrent Program log
- Asset Downtime Process - the log is visible in the Concurrent Program log
- Asset Genealogy Import Process - the log is visible in the Concurrent Program log
- Work Order Business Object API

This profile option is visible and updateable at the User level; it is visible and updateable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: Include Non-Nettable Quantity

If this profile is set to NO, Material Requirements within the Material Requirements, Operations, Work Orders, and the Maintenance Workbench windows display on-hand quantity as the quantity available in Nettable Subinventories. If this profile is set to YES, Material Requirements within these windows display the on-hand quantity as the quantity available in all Subinventories (Nettable and Non-nettable). Within the Maintenance Super User responsibility's Work Orders tab, the Material Requirements page displays he on-hand quantity as the quantity available in Nettable Subinventories if this profile is set to NO. The Material Requirements page displays on-hand quantity as the quantity available in all Subinventories (Nettable and Non-nettable), if the profile is set to YES.

EAM: Maintenance Supervisor

Set this profile option to indicate if the self service Maintenance Supervisor Workbench is enabled.

- *Yes:* The Maintenance Supervisor Workbench is enabled.
- *No:* The Maintenance Supervisor Workbench is not enabled.

EAM: Maintenance Work Request Options

This profile options controls the view and update of a Work Request. It determines whether you can see only your own work requests or also others' work requests.

EAM: Object Instantiation API Log File Directory

The Object Instantiation API is called when a new asset number is created via the Asset Number window or if the Instantiate button is chosen within the Rebuildable Serial Number window. This profile option indicates the debug file directory that the debug

log will reside, if debugging is enabled for the Object Instantiation API. For example, If the profile option, EAM: Object Instantiation API Log Option, is set to Yes, the log of the activity process is saved in the directory determined by the path designated in the current profile option. This profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: Object Instantiation API Log File Name

The Object Instantiation API is called when a new asset number is created via the Asset Number window or if the Instantiate button is chosen within the Rebuildable Serial Number window. This profile option indicates the debug file name that resides in the debug file directory, if debugging is enabled for the Object Instantiation API. For example, If the profile option, EAM: Object Instantiation API Log Option, is set to Yes, the log of the activity process is saved in the directory determined by the path designated by the EAM: Object Instantiation API Log File Directory profile option. The name of the file that includes the activity process log is determined by the current profile option. This profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: Object Instantiation API Log Option

The Object Instantiation API is called when a new asset number is created via the Asset Number window or if the Instantiate button is chosen within the Rebuildable Serial Number window. This profile option indicates whether debugging is enabled for the Activity API. This profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels.

EAM: PM Debug File (including complete file path)

This profile option indicates the directory location where debug files will be created if debug mode is enabled. The profile option is visible and changeable at the User level; it is visible and changeable at System Administrator's User, Responsibility, Application, and Site levels. The internal name of the profile option is EAM_DEBUG_FILE. The default value is a file path inserted by the auto-configuration utility, executed during the installation of the application, and appended by the file name, viz.eampmeng.log.

Profile Options in Other Applications

Profiles that help the eAM integration with other Oracle Applications are listed below. If you are implementing other Oracle Applications, you will need to set up additional profile options as appropriate. Please refer to the applications' respective user's guides or implementation manuals for more details.

Profile Options Summary in Other Applications

Profile Option	User	System Admin: USER	System Admin: RESP	System Admin: APP	System Admin: SITE	Required?	Default
INV: Default Primary Unit of Measure	View Update	View Update	View Update	View Update	View Update	Y	-
INV: Item Default Status	View Update	View Update	View Update	View Update	View Update	Y	-
PO: Enable Direct Delivery To Shop Floor	View Update	View Update	View Update	View Update	View Update	N	-

Profile Option Details

Relevant Profile Options in other applications are listed below:

INV: Default Primary Unit of Measure

You can specify any user-defined or seeded value for this profile.

INV: Item Default Status

You can specify any user-defined or seeded value for this profile.

PO: Enable Direct Delivery To Shop Floor

This profile option is a prerequisite for an eAM User to create Purchase Requisitions for Direct Items via Oracle Purchasing and iProcurement. This profile option must be set to Yes if you need to create Purchase Requisitions using Oracle Purchasing and iProcurement applications.

Related Topics

Personal Profile Values Window, *Oracle Applications User's Guide*

Overview of Setting User Profiles, *Oracle Applications System Administrator's Guide*

Common User Profile Options, *Oracle Applications User's Guide*

19

Oracle Complex Maintenance Repair and Overhaul Open Interfaces

This chapter covers the following topics:

- Complex Maintenance Repair and Overhaul API Overview

Complex Maintenance Repair and Overhaul API Overview

The following parameters are described for Complex Maintenance Repair and Overhaul APIs

Document Index

AHL_DI_DOC_INDEX_PUB. Create_Document

Used to create new document index record and its associated Suppliers and Recipients.

Parameter	Usage	Type	Default
p_x_document_tbl	IN/OUT	DOCUMENT_TBL	-
p_x_supplier_tbl	IN/OUT	SUPPLIER_TBL	-
p_x_recipient_tbl	IN/OUT	RECIPIENT_TBL	-

AHL_DI_DOC_INDEX_PUB. MODIFY_DOCUMENT

Used to update Document.

Parameter	Usage	Type	Default
p_x_document_tbl	IN/OUT	DOCUMENT_TBL	-
p_x_supplier_tbl	IN/OUT	SUPPLIER_TBL	-
p_x_recipient_tbl	IN/OUT	RECIPIENT_TBL	-

AHL_DI_DOC_REVISION_PUB. CREATE_REVISION

Used to create document revisions.

Parameter	Usage	Type	Default
p_x_revision_tbl	IN/OUT	REVISION_TBL	-

AHL_DI_DOC_REVISION_PUB. MODIFY_REVISION

Used to update document revisions.

Parameter	Usage	Type	Default
p_x_revision_tbl	IN/OUT	REVISION_TBL	-

AHL_DI_SUBSCRIPTION_PUB. CREATE_SUBSCRIPTION

Used to create a document subscription.

Parameter	Usage	Type	Default
p_x_subscription_tbl	IN/OUT	SUBSCRIPTION_TBL	-

AHL_DI_SUBSCRIPTION_PUB. MODIFY_SUBSCRIPTION

Used to update a document subscription.

Parameter	Usage	Type	Default
p_x_subscription_tbl	IN/OUT	SUBSCRIPTION_TBL	-

Master Configuration

AHL_MC_MASTERCONFIG_PUB.process_master_config

Used to create and update and delete a Master Configuration root node.

Parameter	Usage	Type	Default
P_x_mc_header_rec	IN/OUT	AHL_MC_MASTERCONFIG_PVT.Header_Rec_Type	-
p_x_node_rec	IN/OUT	AHL_MC_NODE_PVT.Node_Rec_Type	-

AHL_MC_ITEMGROUP_PUB. Process_Item_group

Used to create, update, and delete an Alternate Item Group.

Parameter	Usage	Type	Default
p_x_item_group_rec	IN/OUT	AHL_MC_ItemGroup_PVT.Item_Group_Rec_Type	-
p_x_items_tbl	IN/OUT	AHL_MC_ItemGroup_PVT.Item_Association_Tbl_Type	-

AHL_MC_ITEM_COMP_PUB. Process_Item_Composition

Used to create, update and delete an Item Composition.

Parameter	Usage	Type	Default
p_x_ic_header_rec	IN/OUT	AHL_MC_ITEM_COMP_PVT.Header_Rec_Type	-
p_x_ic_det_tbl	IN/OUT	AHL_MC_ITEM_COMP_PVT.Det_Tbl_Type	-

AHL_MC_NODE_PUB.Process_Node

Used to handle the creation, updating and deletion of a Master Configuration Node.

Parameter	Usage	Type	Default
p_x_node_rec	IN	AHL_MC_Node_Pvt.Node_Rec_Type	-
p_x_counter_rules_tbl	IN	AHL_MC_Node_Pvt.Counter_Rules_Tbl_Type	-
p_x_subconfig_tbl	IN	AHL_MC_Node_Pvt.SubConfig_Tbl_Type	-

AHL_MC_NODE_PUB. DELETE_NODES

Used to delete Master Configuration nodes.

Parameter	Usage	Type	Default
p_nodes_tbl	IN	AHL_MC_Node_Pvt.Node_Tbl_Type	-

Outside Processing

AHL_OSP_ORDERS_PUB.process_osp_order

Used to create, update and delete OSP Orders and lines.

Parameter	Usage	Type	Default
p_x_osp_order_rec	IN/OUT	AHL_OSP_ORDERS_PVT.osp_order_rec_type	-
p_x_osp_order_lines_tbl	IN/OUT	AHL_OSP_ORDERS_PVT.osp_order_lines_tbl_type	-

AHL_OSP_SHIPMENT_PUB. process_order

Used to create/update/delete shipment header and lines associated to an OSP Order and call corresponding CSI sub transaction API.

Parameter	Usage	Type	Default
P_x_header_rec	IN/OUT	AHL_OSP_SHIPMENT_PUB.Ship_header_rec_type	-
P_x_lines_tbl	IN/OUT	AHL_OSP_SHIPMENT_PUB.Ship_lines_tbl_type	-

AHL_OSP_SHIPMENT_PUB. book_order

This API is to book one or more shipment orders corresponding to OSP Orders.

Parameter	Usage	Type	Default
P_oe_header_tbl	IN	Table of NUMBERs	-

AHL_OSP_SHIPMENT_PUB. delete_cancel_order

Used to check if order or line is pre-booking, then delete. If booked but pre-shipping, then cancel. Otherwise, do nothing.

Parameter	Usage	Type	Default
p_oe_header_id	IN	NUMBER	-
P_oe_lines_tbl	IN	Table of NUMBERs	-
P_cancel_flag	IN	VARCHAR2(1)	-

Product Classification

AHL_PC_HEADER_PUB.process_pc_header

Used to create, update delete and copy Product Classification headers.

Parameter	Usage	Type	Default
p_x_pc_header_rec	IN/OUT	AHL_PC_HEADER_PUB.process_pc_header	-

AHL_PC_NODE_PUB.process_nodes

Used to create, update delete and copy Product Classification nodes.

Parameter	Usage	Type	Default
p_x_nodes_tbl	IN/OUT	AHL_PC_NODE_PUB.pc_node_tbl	-

Unit Configuration

AHL_UC_Utilization_PUB.Update_Utilization

Used to update the utilization based on the counter rules defined in the master configuration, given the details of an item/counter id/counter name/uom_code. Also cascades the updates down to all the children if the cascade_flag is set to Y

Parameter	Usage	Type	Default
p_Utilization_tbl	IN	AHL_UC_Utilization_PVT.Utilization_Tbl_Type	-

AHL_UC_UNITCONFIG_PUB.Process_UC_Header

Used to create, update or expire a UC header record in table ahl_unit_config_headers.

Parameter	Usage	Type	Default
p_x_uc_header_rec	IN/OUT	AHL_UC_INSTANCE_PVT.uc_header_rec_type	-

AHL_UC_VALIDATION_PUB.Check_Completeness

Used to check the units completeness and update Complete/Incomplete status if current

status is complete or incomplete.

Parameter	Usage	Type	Default
p_unit_header_id	IN	NUMBER	-
x_evaluation_status	OUT	VARCHAR2	-

AHL_UC_VALIDATION_PUB.Validate_Complete_for_Pos

Used to validate the units (taking the instance_id as a parameter) completeness and checks for all validations.

Parameter	Usage	Type	Default
p_csi_instance_id	IN	NUMBER	-
x_error_tbl	OUT	AHL_UC_VALIDATION_PUB.error_tbl_type	-

AHL_UC_VALIDATION_PUB.Validate_Completeness

Used to validate the units (taking the uc_header_id as a parameter) completeness and checks for all validations.

Parameter	Usage	Type	Default
p_unit_header_id	IN	NUMBER	-
x_error_tbl	OUT	AHL_UC_VALIDATION_PUB.error_tbl_type	-

AHL_UC_INSTANCE_PUB.Unassociate_Instance

Used to nullify a child instances position reference but keep the parent-child relationship in a UC tree structure (in other word, to make the child instance as an extra node in the UC).

Parameter	Usage	Type	Default
p_uc_header_id	IN	NUMBER	NULL
p_uc_name	IN	VARCHAR2	-
p_instance_id	IN	NUMBER	NULL
p_instance_num	IN	VARCHAR2	-
p_prod_user_flag	IN	VARCHAR2	-

AHL_UC_INSTANCE_PUB.Remove_Instance

Used to remove (uninstall) an instance (leaf, branch node or sub-unit) from a UC node. After uninstallation, this instance is available to be reinstalled in another appropriate position.

Parameter	Usage	Type	Default
p_uc_header_id	IN	NUMBER	NULL
p_uc_name	IN	VARCHAR2	-
p_instance_id	IN	NUMBER	NULL
p_instance_num	IN	VARCHAR2	-
p_prod_user_flag	IN	VARCHAR2	-

AHL_UC_INSTANCE_PUB.Update_Instance

Used to update an instances (top node or non top node) attributes (serial number, serial_number_tag, lot_number, revision, mfg_date and etc.)

Parameter	Usage	Type	Default
p_uc_header_id	IN	NUMBER	NULL
p_uc_name	IN	VARCHAR2	-

Parameter	Usage	Type	Default
p_uc_instance_rec	IN	AHL_UC_INSTANCE_PVT.uc_instance_rec_type	-
p_prod_user_flag	IN	VARCHAR2	-

AHL_UC_INSTANCE_PUB.Create_Install_Instance

Used to create a new instance in csi_item_instances and assign it to a UC node.

Parameter	Usage	Type	Default
p_uc_header_id	IN	NUMBER	NULL
p_uc_name	IN	VARCHAR2	-
p_parent_instance_id	IN	NUMBER	NULL
p_parent_instance_num	IN	VARCHAR2	-
p_prod_user_flag	IN	VARCHAR2	-
p_x_uc_instance_rec	IN/OUT	AHL_UC_INSTANCE_PVT.uc_instance_rec_type	-
p_x_sub_uc_rec	IN/OUT	AHL_UC_INSTANCE_PVT.uc_header_rec_type	-
x_warning_msg_tbl	OUT	AHL_UC_VALIDATION_PUB.error_tbl_type	-

AHL_UC_INSTANCE_PUB.Install_Instance

Used to assign an existing instance to a UC node.

Parameter	Usage	Type	Default
p_uc_header_id	IN	NUMBER	NULL
p_uc_name	IN	VARCHAR2	-
p_parent_instance_id	IN	NUMBER	NULL
p_parent_instance_num	IN	VARCHAR2	-
p_instance_id	IN	NUMBER	NULL
p_instance_num	IN	VARCHAR2	-
p_relationship_id	IN	NUMBER	-
p_prod_user_flag	IN	VARCHAR2	-
x_warning_msg_tbl	OUT	AHL_UC_VALIDATION_PUB.error_tbl_type	-

AHL_UC_INSTANCE_PUB.Swap_Instance

Used by production user to make parts change: replace an old instance with a new one in a UC tree.

Parameter	Usage	Type	Default
p_uc_header_id	IN	NUMBER	NULL
p_uc_name	IN	VARCHAR2	-
p_parent_instance_id	IN	NUMBER	NULL
p_parent_instance_num	IN	VARCHAR2	-
p__old_instance_id	IN	NUMBER	NULL
p_old_instance_num	IN	VARCHAR2	-

Parameter	Usage	Type	Default
p__new_instance_id	IN	NUMBER	NULL
p_new_instance_num	IN	VARCHAR2	-
p_relationship_id	IN	NUMBER	-
p_prod_user_flag	IN	VARCHAR2	-
x_warning_msg_tbl	OUT	AHL_UC_VALIDATION_PUB.error_tbl_type	-

Unit Maintenance Plan

AHL_UMP_UF_PUB.process_utilization_forecast

Used to create, update and delete utilization forecast for assets/classifications.

Parameter	Usage	Type	Default
p_x_uf_header_rec	IN/OUT	AHL_UMP_UF_PVT.uf_header_rec_type	-
p_x_uf_details_tbl	IN/OUT	AHL_UMP_UF_PVT.uf_details_tbl_type	-

AHL_UMP_UNITMAINT_PUB.Capture_MR_Updates

For a given set of instance effectivities, this API will record their statuses with either "accomplishment date" or "MR termination date" with their corresponding counter and counter values.

Parameter	Usage	Type	Default
p_unit_Effectivity_tbl	IN	AHL_UMP_UNITMAINT_PVT.Unit_Effectivity_tbl_type	-

Parameter	Usage	Type	Default
p_x_unit_threshold_tbl	IN/OUT	AHL_UMP_UNITMAINT_PVT.Unit_Threshold_tbl_type	-
p_x_unit_accomplish_tbl	IN/OUT	AHL_UMP_UNITMAINT_PVT.Unit_Accomplish_tbl_type	-

AHL_UMP_UNITMAINT_PUB. Process_UnitEffectivity

This API will process all units or a specific unit or a specific MR to build unit effectivity and calculate their due dates.

Parameter	Usage	Type	Default
p_mr_header_id	IN	NUMBER	-
p_mr_titlel	IN	VARCHAR2	-
p_mr_version_number	IN	NUMBER	-
p_unit_config_header_id	IN	NUMBER	-
p_unit_name	IN	VARCHAR2	-
p_csi_item_instance_id	IN	NUMBER	-
p_csi_instance_number	IN	VARCHAR2	-

20

Oracle Service Contracts APIs Introduction

Overview

The public APIs provided by the Oracle Service Contracts application and described in this document are divided into groups of public packages. There are one or more packages for each of the following modules covered here.

Within the appropriate package, each API is specified by listing its code definition, including all of its parameters. The API parameters are then defined. In addition, the data structures used by the individual APIs are also defined, as are the relevant status messages for each API.

> **Note:** The words *procedure* and *API* are used interchangeably in this document.

Parameter Specifications

The specifications for the public APIs provided by the Oracle CRM Applications define four categories of parameters:

- Standard IN
- Standard OUT
- Procedure specific IN
- Procedure specific OUT

Standard IN and OUT parameters are specified by the Oracle Applications business object API Coding Standards, and are discussed in the following sections.

Procedure specific IN and OUT parameter are related to the API being specified, and are discussed with that individual API.

Standard IN Parameters

The following table describes standard IN parameters, which are common to all public APIs provided by Oracle CRM Applications.

Parameter	Data Type	Required	Description
p_api_version	NUMBER	Yes	This must match the version number of the API. An unexpected error is returned if the calling program version number is incompatible with the current API version number (provided in the documentation).
p_init_msg_list	VARCHAR2	Yes	Default = FND_API.G_FALSE: • If set to true, then the API makes a call to *fnd_msg_pub.initialize* to initialize the message stack. • If set to false then the calling program must initialize the message stack. This action is required to be performed only once, even in the case where more than one API is called.

Parameter	Data Type	Required	Description
p_commit	VARCHAR2(1)	No	Default = FND_API.G_FALSE: • If set to true, then the API commits before returning to the calling program. • If set to false, then it is the calling program's responsibility to commit the transaction.

Standard OUT Parameters

The following table describes standard OUT parameters, which are common to all public APIs provided by Oracle CRM Applications.

Note: All standard OUT parameters are required.

Parameter	Data Type	Description
x_return_status	VARCHAR2(1)	Indicates the return status of the API. The values returned are one of the following: • FND_API.G_RET_STS_SUCCESS Success: Indicates the API call was successful • FND_API.G_RET_STS_ERROR Expected Error: There is a validation error, or missing data error. • FND_API.G_RET_STS_UNEXP_ERROR Unexpected Error: The calling program can not correct the error.
x_msg_count	NUMBER	Holds the number of messages in the message list.
x_msg_data	VARCHAR2(2000)	Holds the encoded message if *x_msg_count* is equal to one.

Parameter Size

Verify the size of the column from the base table for that column when passing a parameter of a specific length. For example, if you pass a NUMBER value, first query to find the exact value to pass. An incorrect value can cause the API call to fail.

Missing Parameter Attributes

The following table describes optional IN parameters which are initialized to pre-defined values representing missing constants. These constants are defined for the common PL/SQL data types and should be used in the initialization of the API formal parameters.

Parameter	Type	Initialized Value
g_miss_num	CONSTANT	NUMBER:= 9.99E125
g_miss_char	CONSTANT	VARCHAR2(1):= chr(0)
g_miss_date	CONSTANT	DATE:= TO_DATE('1','j');

These constants are defined in the package FND_API in the file *fndpapis.pls*. All columns in a record definition are set to the G_MISS_X constant as defined for the data type.

Parameter Validations

The following types of parameters are always validated during the API call:

- Standard IN
- Standard OUT
- Mandatory procedure specific IN
- Procedure specific OUT

Invalid Parameters

If the API encounters any invalid parameters during the API call, then one of the following actions will occur:

- An exception is raised.
- An error message identifying the invalid parameter is generated.
- All API actions are cancelled.

Version Information

It is mandatory that every API call pass a version number for that API as its first parameter (*p_api_version*).

This version number must match the internal version number of that API. An unexpected error is returned if the calling program version number is incompatible with the current API version number.

Warning: The currently supported version at this time is 1.0 Use only this for the API version number.

In addition, the object version number **must** be input for all update and delete APIs.

- If the *object_version_number* passed by the API matches that of the object in the database, then the update is completed.
- If the *object_version_number* passed by the API does not match that of the object in the database, then an error condition is generated.

Status Messages

Note: It is not required that all status notifications provide a number identifier along with the message, although, in many cases, it is provided.

Every API must return one of the following states as parameter *x_return_status* after the API is called:

- S (Success)
- E (Error)
- U (Unexpected error)

Each state can be associated with a status message. The following table describes each state.

Status	Description
S	Indicates that the API performed all the operations requested by its caller. • A success return status may or may not be accompanied by messages in the API message list. • Currently, the CRM APIs do not provide a message for a return status of success.

Status	Description
E	Indicates that the API failed to perform one or more of the operations requested by its caller. An error return status is accompanied by one or more messages describing the error.
U	Indicates that the API encountered an error condition it did not expect, or could not handle, and that it is unable to continue with its regular processing. • For example, certain programming errors such as attempting to a division by zero will cause this error. • These types of errors usually cannot be corrected by the user and requires a system administrator or application developer to correct.

Warning and Information Messages

In addition to these three types of possible status messages, you can also code the following additional message types:

- Warnings
- Information

To create a warning message, perform the following steps:

Create a global variable to be used to signal a warning condition. For example, this could be similar to the following:

```
G_RET_STS_WARNING := 'W'
```

- This global variable is not part of the FND_API package.
- Return this value if the warning condition is encountered. For example, using the same example as in step one, set up the following code in the API to process the warning condition:

  ```
  x_return_status := G_RET_STS_WARNING
  ```

  This code replaces the more usual:

  ```
  x_return_status := fnd_api.g_ret_sts_unexp_error for "U"
  ```

- If desired, perform a similar procedure to create Information messages.

21

Oracle Asset Tracking API

This chapter covers the following topics:

- Overview of the Oracle Asset Tracking API
- Oracle Asset Tracking Public Package
- Contents of Package CSE_DEPLOYMENT_GRP

Overview of the Oracle Asset Tracking API

This chapter describes the Deployment Transaction APIs for Oracle Asset Tracking.

You can use external deployment systems (such as bar code readers and scanners) along with Oracle Asset Tracking. Oracle Asset Tracking has the ability to integrate with the external systems through these APIs. Call these APIs to update Oracle Asset Tracking when the item is installed, uninstalled, placed in service, moved, or retired.

The topics in this section are as follows:

- Oracle Asset Tracking Public Package, page 21-1
- Contents of Package CSE_DEPLOYMENT_GRP, page 21-4

Oracle Asset Tracking Public Package

The APIs provided for Oracle Asset Tracking are organized into the following package:

- CSE_DEPLOYMENT_GRP

This package contains the procedure PROCESS_TRANSACTION, which can perform the deployment transactions that are listed in the following table:

Name	Description
Install	The prerequisite is that you must have an instance with the operational status code In-Process/Not Used for the item you are trying to install. This procedure creates/updates the item instance with the operational status Installed, location type code from Project/Internal Site to HZ_locations. If there is an existing instance with operational status code Installed, then it updates the same or otherwise it will create a new instance.
Un-Install	The prerequisite is that you must have an instance with the operational status code Installed for the item you are trying to uninstall. This procedure updates the item instance with operational status code Not Used, location type code from HZ_locations to Project.
Put in Service	The prerequisite is that you must have an instance with the operational status code Installed for the item you are trying to put into service. This procedure creates/updates the item instance with operational status code In-Service. If there is an existing instance with operational status code In-Service, then it updates the same or otherwise it will create a new instance. This procedure creates expenditure items in the projects interface table. The installed base transaction created by this action will be used to create a project asset for Normal Items.

Name	Description
Take Out of Service	The prerequisite is that you must have an instance with the operational status code In-Service for the item you are trying to put out of service. This procedure creates/updates the item instance with operational status code Out-of-Service. If there is an existing instance with operational status code Out-of-Service, then it updates the same or otherwise it will create a new instance.
Move	The prerequisite is that you must have an instance with the operational status code Out-of-Service for the item you are trying to move. This procedure creates/updates the network location of the item instance with operational status code Out-of-Service. If there is an existing instance with operational status code Out-of-Service, network location as From Network Location, then it updates the network location with To Network Location or otherwise it will create a new instance with usage code Out-of-Service, network location as To Network Location. The installed base transaction created by this action will be used to interface move transactions to Oracle Fixed Assets.
Project Transfer	The prerequisite is that project transfer cannot be done for items that are already put in service. This procedure transfer items that are received into a project location to another project location. Item instances that are just received in to project or Installed in a location can be transferred.

Name	Description
Retirement	This procedure manually retires items that are capitalized. You can retire an item instance operationally (the instance is expired), or both operationally and functionally (the corresponding fixed asset is also retired from the books).

Contents of Package CSE_DEPLOYMENT_GRP

Contains one single routine that handles all of the previously mentioned deployment actions.

PROCESS_TRANSACTION

The following table describes the IN parameters associated with this API.

IN Parameters:

Parameter	Data Type	Required	Description
p_instance_tbl	cse_deployment_grp.txn_instances_tbl	Yes	This PL/SQL structure contains the instance information for the transaction in context.
p_dest_location_tbl	cse_deployment_grp.dest_location_tbl	Yes	This PL/SQL structure contains the destination location information for the item instance being transacted. The location information should be populated in accordance with the transaction action in context.

Parameter	Data Type	Required	Description
p_ext_attrib_values_tbl	cse_deployment_grp.txn_ext_attrib_values_tbl	Yes	This PL/SQL structure contains any references to extended attribute values that require an update along with the instance update.
p_txn_tbl	cse_deployment_grp.transaction_tbl	Yes	This PL/SQL structure holds the transaction action type and the transaction reference data that needed to be populated in install base transactions.

The following table describes the OUT parameters associated with this API.

OUT Parameters:

Parameter	Data Type	Description
x_return_status	VARCHAR2	Returns the status of the transaction: FND_API.G_RET_STS_SUCCESS - successful FND_API.G_RET_STS_ERROR - error FND_API.G_RET_STS_UNEXP_ERROR - unexpected error
x_error_msg	VARCHAR2	The error message

Sample Code

Project Transfer

```
set serverout on
declare

  l_location_type_code      varchar2(30);
  l_location_id             number;
  l_instance_usage_code     varchar2(30);
  l_operational_status_code varchar2(30);

  l_instance_tbl            cse_deployment_grp.txn_instances_tbl;
  l_dest_location_tbl       cse_deployment_grp.dest_location_tbl;
  l_ext_attrib_values_tbl
cse_deployment_grp.txn_ext_attrib_values_tbl;
  l_txn_tbl                 cse_deployment_grp.transaction_tbl;
  l_return_status           varchar2(1) := fnd_api.g_ret_sts_success;
  l_error_message           varchar2(2000);

begin

  l_instance_tbl(1).instance_id                 := 1302711;

  select serial_number,
         lot_number,
         inventory_item_id,
         operational_status_code,
         location_type_code,
         location_id,
         instance_usage_code,
         operational_status_code
  into   l_instance_tbl(1).serial_number,
         l_instance_tbl(1).lot_number,
         l_instance_tbl(1).inventory_item_id,
         l_instance_tbl(1).operational_status_code,
         l_location_type_code,
         l_location_id,
         l_instance_usage_code,
         l_operational_status_code
  from   csi_item_instances
  where  instance_id = l_instance_tbl(1).instance_id;

  IF l_location_type_code = 'PROJECT' THEN
    l_instance_tbl(1).last_pa_project_id          := 5773;
    l_instance_tbl(1).last_pa_project_task_id     := 248906;
  ELSE
    l_instance_tbl(1).last_pa_project_id          := 5773;
    l_instance_tbl(1).last_pa_project_task_id     := 248906;
  END IF;

  l_txn_tbl(1).transaction_id                   := fnd_api.g_miss_num;
  l_txn_tbl(1).transaction_date                 := sysdate;
  l_txn_tbl(1).source_transaction_date          := sysdate;
  l_txn_tbl(1).transaction_type_id              := 152;
  l_txn_tbl(1).txn_sub_type_id                  := 3;
  l_txn_tbl(1).source_group_ref_id              := fnd_api.g_miss_num;
  l_txn_tbl(1).source_group_ref                 := fnd_api.g_miss_char;
  l_txn_tbl(1).source_header_ref_id             := fnd_api.g_miss_num;
  l_txn_tbl(1).source_header_ref                := fnd_api.g_miss_char;
  l_txn_tbl(1).transacted_by                    := fnd_api.g_miss_num;
  l_txn_tbl(1).transaction_quantity             := 1;
  l_txn_tbl(1).operational_flag                 := 'Y';
  l_txn_tbl(1).financial_flag                   := 'Y';
```

```
l_dest_location_tbl(1).parent_tbl_index        := 1;
  l_dest_location_tbl(1).location_type_code      :=
l_location_type_code;
  l_dest_location_tbl(1).location_id             := l_location_id;
  l_dest_location_tbl(1).instance_usage_code     :=
l_instance_usage_code;
  l_dest_location_tbl(1).operational_status_code :=
l_operational_status_code;

  IF l_location_type_code = 'PROJECT' THEN
    l_dest_location_tbl(1).pa_project_id           := 8093;
    l_dest_location_tbl(1).pa_project_task_id      := 249903;
    l_dest_location_tbl(1).last_pa_project_id      :=
fnd_api.g_miss_num;
    l_dest_location_tbl(1).last_pa_project_task_id :=
fnd_api.g_miss_num;
  ELSE
    l_dest_location_tbl(1).last_pa_project_id      := 8093;
    l_dest_location_tbl(1).last_pa_project_task_id := 249903;
    l_dest_location_tbl(1).pa_project_id           :=
fnd_api.g_miss_num;
    l_dest_location_tbl(1).pa_project_task_id      :=
fnd_api.g_miss_num;
  END IF;

  cse_deployment_grp.process_transaction (
    p_instance_tbl            => l_instance_tbl,
    p_dest_location_tbl       => l_dest_location_tbl,
    p_ext_attrib_values_tbl   => l_ext_attrib_values_tbl,
    p_txn_tbl                 => l_txn_tbl,
    x_return_status           => l_return_status,
    x_error_msg               => l_error_message);

  if l_return_status <> fnd_api.g_ret_sts_success then
    dbms_output.put_line(l_error_message);
  end if;

end;
/
```

Asset Retirement

```
set serverout on
declare

  l_instance_tbl            cse_deployment_grp.txn_instances_tbl;
  l_dest_location_tbl       cse_deployment_grp.dest_location_tbl;
  l_ext_attrib_values_tbl
cse_deployment_grp.txn_ext_attrib_values_tbl;
  l_txn_tbl                 cse_deployment_grp.transaction_tbl;
  l_return_status           varchar2(1) := fnd_api.g_ret_sts_success;
  l_error_message           varchar2(2000);

begin

  l_instance_tbl(1).instance_id                 := 1605848;
  l_instance_tbl(1).asset_id                    := 108431;

  select serial_number,
         inventory_item_id,
         operational_status_code
  into   l_instance_tbl(1).serial_number,
         l_instance_tbl(1).inventory_item_id,
         l_instance_tbl(1).operational_status_code
  from   csi_item_instances
  where  instance_id = l_instance_tbl(1).instance_id;

  l_instance_tbl(1).lot_number                  := fnd_api.g_miss_char;
  l_instance_tbl(1).inventory_revision          := fnd_api.g_miss_char;
  l_instance_tbl(1).last_pa_project_id          := fnd_api.g_miss_num;
  l_instance_tbl(1).last_pa_project_task_id     := fnd_api.g_miss_num;
  l_instance_tbl(1).unit_of_measure             := fnd_api.g_miss_char;
  l_instance_tbl(1).active_start_date           := fnd_api.g_miss_date;
  l_instance_tbl(1).active_end_date             := fnd_api.g_miss_date;
  l_instance_tbl(1).instance_status_id          := fnd_api.g_miss_num;

  l_txn_tbl(1).transaction_id                   := fnd_api.g_miss_num;
  l_txn_tbl(1).transaction_date                 := sysdate;
  l_txn_tbl(1).source_transaction_date          := sysdate;
  l_txn_tbl(1).transaction_type_id              := 104;
  l_txn_tbl(1).txn_sub_type_id                  := fnd_api.g_miss_num;
  l_txn_tbl(1).source_group_ref_id              := fnd_api.g_miss_num;
  l_txn_tbl(1).source_group_ref                 := fnd_api.g_miss_char;
  l_txn_tbl(1).source_header_ref_id             := fnd_api.g_miss_num;
  l_txn_tbl(1).source_header_ref                := fnd_api.g_miss_char;
  l_txn_tbl(1).transacted_by                    := fnd_api.g_miss_num;
  l_txn_tbl(1).transaction_quantity             := 1;
  l_txn_tbl(1).proceeds_of_sale                 := 10;
  l_txn_tbl(1).cost_of_removal                  := 10;

  -- use this flag to expire the item instance
  l_txn_tbl(1).operational_flag                 := 'Y';

  -- use this flag to retire the fixed asset
  l_txn_tbl(1).financial_flag                   := 'Y';

  cse_deployment_grp.process_transaction (
    p_instance_tbl            => l_instance_tbl,
    p_dest_location_tbl       => l_dest_location_tbl,
    p_ext_attrib_values_tbl   => l_ext_attrib_values_tbl,
    p_txn_tbl                 => l_txn_tbl,
    x_return_status           => l_return_status,
    x_error_msg               => l_error_message);
```

```
  if l_return_status <> fnd_api.g_ret_sts_success then
    dbms_output.put_line(l_error_message);
  end if;

end;
/
```

22

Oracle Service Contracts Public APIs

Entitlement/OM Integration APIs

Entitlements refers to various services a customer is entitled once a service contract is in effect. A set of public APIs are provided to expose the entitlements to other modules such as Oracle Customer Support, Service Core, Field Service, Depot Repair, andCharges. Any other modules are also welcome to use these APIs within its capabilities explained below. They are query only APIs.

Service Contracts OM/Integration Public API contains various procedures used by Order Management application to get service duration, Check the availability of Service for a customer or a product, get a list of Services available for a Customer or a product.

All the Entitlements OM/Integration APIs procedures are defined in the following packages.

- OKS_ENTITLEMENTS_PUB
- OKS_CON_COVERAGE_PUB
- OKS_OMINT_PUB

OKS_ENTITLEMENTS_PUB - Procedure	Description
Check Coverage Times	This API returns whether a particular date and time in a time zone (may be the sysdate), is covered in the coverage time specified for a given service line and business process. The output may be 'Y' or 'N' (Yes/No).

OKS_ENTITLEMENTS_PUB - Procedure	Description
Check Reaction Times	This API returns the reaction time, unit of measure and react by date (date & time) in a time zone for a given service line and business process. The output may be 2, Hrs, 01-JUN-2001 14:00
Get React Resolve By Time	This API returns the reaction and/or resolution time information in a time zone for a given service line and business process. There are options to get the first or best reaction time, resolution time or both.
Get All Contracts	This API returns the contract header information for any combination of input parameter as explained in API Signature section.
Get Contract Details	This API returns the contract line information for any combination of input parameter as explained in API Signature section.
Get Contracts	This is an over loaded API which returns contract information for different combination of Service, Extended Warranty or Warranty, Coverage Levels and Business Processes. Detailed explanation of input and output parameters are explained in the API signature section of this document.
Get Coverage Levels	This API returns the Coverage Level such as Party, Customer, Site, System, Item and Product information for a Service, Extended Warranty or Warranty.
Get Contacts	This API returns the Contact information for a contract or a line.
Get Preferred Engineers	This API returns the details of Preferred Engineers for a Service, Extended Warranty or Warranty.

OKS_ENTITLEMENTS_PUB - Procedure	Description
Get Coverage Type	This API returns the Coverage type and importance level information for a contract line id.
Get Cov Txn Groups	This API returns the business process line level information in the coverage, based on the business process setup for a given contract line id.
Get Txn Billing Types	For a given transaction group line id, this API returns the information for service activity billing type line and labor bill rate.
Get Contracts Expiration	For a given contract id, this API returns the expiration details including: contract end date, contract grace period, and duration.
Validate Contract Line	This API returns if the contract line id is valid for the given input of covered level table of records and the business process id.

OKS_CON_COVERAGE_PUB - Procedure	Description
Apply Contract Coverage	This API returns the discounted amount for a Service or Extended Warranty based up on the Discount Amount and % Covered, specified in the Service Contract coverage terms for a specific business process.
Get BP Pricelist	This API returns the price list and discount at the coverage business process level and the price list at the contract header level for a contract_line_id, business_process_id and request date.

OKS_OMINT_PUB - Procedure	Description
Get Duration	This API calculates the Service duration based on the cotermination flag and the minimum service duration
Is Service Available	This API checks for the availability of the service for a Customer and a product
Available Services	This API gives the list of services available to a Customer and a Product
OKS Available Services	This API gives the list of services with Name, Description and Coverage Associated available to a Customer and a Product.

Package OKS_ENTITLEMENTS_PUB

The Entitlements APIs consist of the following procedures:

- Check Coverage Times, page 22-5
- Check Reaction Times, page 22-6
- Get React Resolve By Time, page 22-8
- Get All Contracts, page 22-12
- Get Contract Details, page 22-17
- Get Contracts (Overloaded)
  - Get Contracts, page 22-20
  - Get Contracts, page 22-24
  - Get Contracts, page 22-30
- Get Coverage Levels, page 22-35
- Get Contacts, page 22-40
- Get Preferred Engineers, page 22-42

- Get Coverage Type, page 22-44
- Get Cov Txn Groups, page 22-46
- Get Txn Billing Types, page 22-49
- Get Contracts Expiration, page 22-54
- Validate Contract line, page 22-55

Check Coverage Times

This API returns whether a particular date and time in a time zone (may be the sysdate), is covered in the coverage time specified for a given service line and business process. The output may be 'Y' or 'N' (Yes/No).

Procedure Specification

```
PROCEDURE check_coverage_times
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_business_process_id  IN  Number
,p_request_date         IN  Date
,p_time_zone_id         IN  Number
,p_contract_line_id     IN  Number
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_covered_yn           OUT NOCOPY Varchar2);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter

Parameter	Data Type	Required	Description and Validations
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_business_process_id	NUMBER	Yes	Business Process ID
p_request_date	DATE	Yes	Request Date and Time
p_time_zone_id	NUMBER	Yes	Request Time Zone ID
p_contract_line_id	NUMBER	Yes	Line ID of Service, Extended Warranty or Warranty

The following table describes the OUT parameters associated with this API:

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_covered_yn	VARCHAR2	Y/N

Check Reaction Times

This API returns the reaction time, unit of measure and react by date (date & time) in a time zone for a given service line and business process. The output may be 2, Hrs, 01-JUN-2003 14:00.

Procedure Specification

```
PROCEDURE check_reaction_times
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
```

```
,p_business_process_id  IN  Number
,p_request_date         IN  Date
,p_sr_severity          IN  Number
,p_time_zone_id         IN  Number
,p_contract_line_id     IN  Number
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_react_within         OUT NOCOPY Number
,x_react_tuom           OUT NOCOPY Varchar2
,x_react_by_date        OUT NOCOPY Date);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_business_process_id	NUMBER	Yes	Business Process ID
p_request_date	DATE	Yes	Request Date
p_sr_severity	NUMBER	Yes	Severity ID
p_time_zone_id	NUMBER	Yes	Request Time Zone ID
p_contract_line_id	NUMBER	Yes	Line ID of Service, Extended Warranty or Warranty

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_react_within	NUMBER	Reaction Time
x_react_tuom	VARCHAR2	Unit of Measure for Reaction Time
x_react_by_date	DATE	Date and Time by which the reaction or response has to be made for a Service Request.

Get React Resolve By Time

This API returns the reaction and/or resolution time information in a time zone for a given service line. There are options to get the first or best reaction time, resolution time or both.

Procedure Specification

```
PROCEDURE get_react_resolve_by_time
(p_api_version              in  number
,p_init_msg_list            in  varchar2
,p_inp_rec                  in  grt_inp_rec_type
,x_return_status            out nocopy varchar2
,x_msg_count                out nocopy number
,x_msg_data                 out nocopy varchar2
,x_react_rec                out rcn_rsn_rec_type
,x_resolve_rec              out rcn_rsn_rec_type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_inp_rec	grt_inp_rec_type	Yes	See the Data Structure Specification: grt_inp_rec_type

grt_inp_rec_type

Record Specification

```
TYPE grt_inp_rec_type IS RECORD
(contract_line_id     number
,business_process_id  okx_bus_processes_v.id1%type
,severity_id          okx_incident_severits_v.id1%type
,request_date         date
,time_zone_id         okx_timezones_v.timezone_id%type
,category_rcn_rsn     okc_rules_b.rule_information_category%type
,compute_option       varchar2(10));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description and Validations
contract_line_id	NUMBER	Yes	Contract Line ID
business_process_id	NUMBER	Yes	Business Process ID
severity_id	NUMBER	Yes	Severity ID
request_date	DATE	No	Request Date. The default is system date.

Parameter	Data Type	Required	Description and Validations
time_zone_id	NUMBER	Yes	Request Time Zone ID
category_rcn_rsn	VARCHAR2	Yes	OKS_ENTITLEMENTS_PUB.G_REACTION - Returns reaction time information OKS_ENTITLEMENTS_PUB.G_RESOLUTION - Returns resolution time information OKS_ENTITLEMENTS_PUB.G_REACT_RESOLVE - Returns reaction and resolution time information
compute_option	VARCHAR2	Yes	OKS_ENTITLEMENTS_PUB.G_BEST - Returns the best reaction and/or resolution time information OKS_ENTITLEMENTS_PUB.G_FIRST - Returns the first reaction and/or resolution time information

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter

Parameter	Data Type	Description
x_msg_data	VARCHAR2	Standard OUT Parameter
x_react_rec	rcn_rsn_rec_type	Reaction Time information. See the Data Structure Specification: rcn_rsn_rec_type
x_resolve_rec	rcn_rsn_rec_type	Resolution Time information. See the Data Structure Specification: rcn_rsn_rec_type

rcn_rsn_rec_type

Record Specification

```
TYPE rcn_rsn_rec_type IS RECORD
(duration            okc_react_intervals.duration%type
,uom                 okc_react_intervals.uom_code%type
,by_date_start       date
,by_date_end         date);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
duration	NUMBER	Reaction or Resolution Time
uom	VARCHAR2	Unit of Measure for Reaction or Resolution Time
by_date_start	DATE	Date and Time by which the Reaction / Response or Resolution has begun for a Service Request.

Parameter	Data Type	Description
by_date_end	DATE	Date and Time by which the Reaction / Response or Resolution has to be completed for a Service Request.

Get All Contracts

This API returns the contract header information for any combination of input parameter as explained in API Signature section.

Procedure Specification

```
PROCEDURE get_all_contracts
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_inp_rec              IN  inp_rec_type
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_all_contracts        OUT NOCOPY hdr_tbl_type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter

Parameter	Data Type	Required	Description and Validations
p_inp_rec	inp_rec_type	Yes	See the Data Structure Specification: inp_rec_type

inp_rec_type

Record Specification

```
TYPE inp_rec_type IS RECORD
(contract_id               NUMBER
,contract_status_code      VARCHAR2(30)
,contract_type_code        VARCHAR2(30)
,end_date_active           DATE
,party_id                  NUMBER);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
contract_id	NUMBER	Yes	Contract Header ID
contract_status_code	VARCHAR2	No	Contract Status Code
contract_type_code	VARCHAR2	No	Contract Type Code
end_date_active	DATE	No	End Date Active
party_id	NUMBER	No	Party ID

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter

Parameter	Data Type	Description
x_msg_data	VARCHAR2	Standard OUT Parameter
x_all_contracts	hdr_tbl_type	Contract header information. See the Data Structure Specification: hdr_tbl_type

hdr_tbl_type

Record Specification

```
TYPE hdr_tbl_type IS TABLE OF hdr_rec_type
INDEX BY BINARY_INTEGER;
TYPE hdr_rec_type IS RECORD
(ORG_ID                  OKC_K_HEADERS_B.AUTHORING_ORG_ID%TYPE
,CONTRACT_ID             OKC_K_HEADERS_B.ID%TYPE
,CONTRACT_NUMBER         OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,SHORT_DESCRIPTION       KC_K_HEADERS_TL.SHORT_DESCRIPTION%TYPE
,CONTRACT_AMOUNT         Number(18,2)
,CONTRACT_STATUS_CODE    OKC_K_HEADERS_B.STS_CODE%TYPE
,CONTRACT_TYPE           OKC_K_HEADERS_B.CHR_TYPE%TYPE
,PARTY_ID                Number
,TEMPLATE_YN             OKC_K_HEADERS_B.TEMPLATE_YN%TYPE
,TEMPLATE_USED           OKC_K_HEADERS_B.TEMPLATE_USED%TYPE
,DURATION                Number
,PERIOD_CODE             Varchar2(25)
,START_DATE_ACTIVE       OKC_K_HEADERS_B.START_DATE%TYPE
,END_DATE_ACTIVE         OKC_K_HEADERS_B.END_DATE%TYPE
,BILL_TO_SITE_USE_ID     Number
,SHIP_TO_SITE_USE_ID     Number
,AGREEMENT_ID            OKC_K_HEADERS_B.CHR_ID_AWARD%TYPE
,PRICE_LIST_ID           Number
,MODIFIER                Number
,CURRENCY_CODE           Varchar2(25)
,ACCOUNTING_RULE_ID      Number
,INVOICING_RULE_ID       Number
,TERMS_ID                Number
,PO_NUMBER               OKC_K_HEADERS_B.CUST_PO_NUMBER%TYPE
```

```
,BILLING_PROFILE_ID      Number
,BILLING_FREQUENCY       Varchar2(25)
,BILLING_METHOD          Varchar2(3)
,REGULAR_OFFSET_DAYS     Number
,FIRST_BILL_TO           Date
,FIRST_BILL_ON           Date
,AUTO_RENEW_BEFORE_DAYS OKC_K_HEADERS_B.AUTO_RENEW_DAYS%TYPE
,QA_CHECK_LIST_ID        OKC_K_HEADERS_B.QCL_ID%TYPE
,RENEWAL_NOTE            CLOB
,TERMINATION_NOTE        CLOB
,TAX_EXEMPTION           Varchar2(450)
,TAX_STATUS              Varchar2(450)
,CONVERSION_TYPE         Varchar2(450));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
org_id	NUMBER	Org ID
contract_id	NUMBER	Contract Header ID
contract_number	VARCHAR2	Contract Number
short_description	VARCHAR2	Short Description
contract_amount	NUMBER	Contract Amount
contract_status_code	VARCHAR2	Contract Status Code
contract_type	VARCHAR2	Contract Type
party_id	NUMBER	Party ID
template_yn	VARCHAR2	Y/N
template_used	VARCHAR2	Contract Template Name
duration	NUMBER	Contract Duration

Parameter	Data Type	Description
period_code	NUMBER	Period Code
start_date_active	DATE	Start Date Active
end_date_active	DATE	End Date Active
bill_to_site_use_id	NUMBER	Bill To Site Use ID
ship_to_site_use_id	NUMBER	Ship To Site Use ID
agreement_id	NUMBER	Agreement ID
price_list_id	NUMBER	Price List ID
modifier	NUMBER	Contract Number Modifier
currency_code	VARCHAR2	Currency Code
accounting_rule_id	NUMBER	Accounting Rule ID
invoicing_rule_id	NUMBER	Invoicing Rule ID
terms_id	NUMBER	Terms ID
po_number	VARCHAR2	Purchase Order Number
billing_profile_id	NUMBER	Billing Profile ID
billing_frequency	VARCHAR2	Billing Frequency
billing_method	VARCHAR2	Billing Method
regular_offset_days	NUMBER	Regular Offset Days
first_bill_to	DATE	First Bill To Date
first_bill_on	DATE	First Bill On Date
auto_renew_before_days	NUMBER	Auto Renew Before Days

Parameter	Data Type	Description
qa_check_list_id	NUMBER	QA Check List ID
renewal_note	CLOB	Renewal Note
termination_note	CLOB	Termination Note
tax_exemption	VARCHAR2	Tax Exemption
tax_status	VARCHAR2	Tax Status
conversion_type	VARCHAR2	Conversion Type

Get Contract Details

This API returns the contract line information for any combination of input parameter as explained in API Signature section.

Procedure Specification

```
PROCEDURE get_contract_details
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_contract_line_id     IN  Number
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_all_lines            OUT NOCOPY line_tbl_type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_contract_line_id	NUMBER	Yes	Contract Line ID

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_all_lines	line_tbl_type	Contract line information. See the Data Structure Specification: line_tbl_type

line_tbl_type

Record Specification

```
TYPE line_tbl_type IS TABLE OF line_rec_type
INDEX BY BINARY_INTEGER;
TYPE line_rec_type IS RECORD
(CONTRACT_LINE_ID          OKC_K_LINES_B.ID%TYPE
,COnTRACT_PARENT_LINE_ID  OKC_K_LINES_B.CLE_ID%TYPE
,CONTRACT_ID              OKC_K_LINES_B.CHR_ID%TYPE
,LINE_STATUS_CODE         OKC_K_LINES_B.STS_CODE%TYPE
,DURATION                 Number
,PERIOD_CODE              Varchar2(25)
,START_DATE_ACTIVE        OKC_K_HEADERS_B.START_DATE%TYPE
,END_DATE_ACTIVE          OKC_K_HEADERS_B.END_DATE%TYPE
,LINE_NAME                Varchar2(150)
```

```
,BILL_TO_SITE_USE_ID      Number
,SHIP_TO_SITE_USE_ID      Number
,AGREEMENT_ID             OKC_K_HEADERS_B.CHR_ID_AWARD%TYPE
,MODIFIER                 Number
,PRICE_LIST_ID            Number
,PRICE_NEGOTIATED         OKC_K_LINES_B.PRICE_NEGOTIATED%TYPE
,BILLING_PROFILE_ID       Number
,BILLING_FREQUENCY        Varchar2(25)
,BILLING_METHOD           Varchar2(3)
,REGULAR_OFFSET_DAYS      Number
,FIRST_BILL_TO            Date
,FIRST_BILL_ON            Date
,TERMINATION_DATE         Date);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
contract_line_id	NUMBER	Contract Line ID
contract_parent_line_id	NUMBER	Contract Parent Line ID
contract_id	NUMBER	Contract Header ID
line_status_code	VARCHAR2	Line Status Code
duration	NUMBER	Duration
period_code	VARCHAR2	Period Code
start_date_active	DATE	Start Date Active
end_date_active	DATE	End Date Active
line_name	VARCHAR2	Line Name
bill_to_site_use_id	NUMBER	Bill To Site Use Id
ship_to_site_use_id	NUMBER	Ship To Site Use Id

Parameter	Data Type	Description
agreement_id	NUMBER	Agreement Id
modifier	NUMBER	Modifier
price_list_id	NUMBER	Price List Id
price_negotiated	NUMBER	Price Negotiated
billing_profile_id	NUMBER	Billing Profile Id
billing_frequency	VARCHAR2	Billing Frequency
billing_method	VARCHAR2	Billing Method
regular_offset_days	NUMBER	Regular Offset Days
first_bill_to	DATE	First Bill To
first_bill_on	DATE	First Bill On
termination_date	DATE	Termination Date

Get Contracts

This is an over loaded API which returns contract information for different Coverage Levels such as Party, Customer, Site, System, Item, and Product. If the input parameter validate_flag is set to `Y', request date is checked against the Date Effectivity and only those contract covered level lines eligible for entitlements are returned. This API also returns the coverage type and associated importance level information.

Procedure Specification

```
PROCEDURE get_contracts
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_inp_rec              IN  inp_cont_rec
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
```

```
,x_ent_contracts        OUT NOCOPY ent_cont_tbl);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_inp_rec	inp_cont_rec	Yes	Input record. See the Data Structure Specification: inp_cont_rec

inp_cont_rec

Record Specification

```
TYPE inp_cont_rec IS RECORD
(contract_number          OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,coverage_level_line_id  Number
,party_id                Number
,site_id                 Number
,cust_acct_id            Number
,system_id               Number
,item_id                 Number
,product_id              Number
,request_date            Date
,validate_flag           Varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
contract_number	VARCHAR2	No	Contract Number
coverage_level_line_id	NUMBER	No	Coverage Level Line Id
party_id	NUMBER	No	Party Id
site_id	NUMBER	No	Site Id
cust_acct_id	NUMBER	No	Cust Acct Id
system_id	NUMBER	No	System Id
item_id	NUMBER	No	Item Id
product_id	NUMBER	No	Product Id
request_date	DATE	No	Request Date. The Default is sysdate.
validate_flag	VARCHAR2	No	Validate Flag Y/N. Valid values are 'Y' or, 'N'. Default is 'N'. If 'Y' is passed as input only Valid records are returned.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_ent_contracts	ent_cont_tbl	Contract information. See the Data Structure Specification: ent_cont_tbl

ent_cont_tbl

Record Specification

```
TYPE ent_cont_tbl IS TABLE OF ent_cont_rec
INDEX BY BINARY_INTEGER;
TYPE ent_cont_rec IS RECORD
(contract_id                Number
,contract_number           OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,service_line_id           Number
,service_name              OKC_K_LINES_V.NAME%TYPE
,service_description       OKC_K_LINES_V.ITEM_DESCRIPTION%TYPE
,coverage_term_line_id     Number
,Coverage_term_name        OKC_K_LINES_V.NAME%TYPE
,coverage_term_description OKC_K_LINES_V.ITEM_DESCRIPTION%TYPE
,coverage_level_line_id    Number
,coverage_level            OKC_LINE_STYLES_TL.NAME%TYPE
,coverage_level_code       OKC_LINE_STYLES_B.LTY_CODE%TYPE
,coverage_level_start_date Date
,coverage_level_End_date   Date
,coverage_level_id         Number
,warranty_flag             Varchar2(1)
,eligible_for_entitlement  Varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
contract_id	NUMBER	Contract Id
contract_number	VARCHAR2	Contract Number
service_line_id	NUMBER	Service Line Id
service_name	VARCHAR2	Service Name
service_description	VARCHAR2	Service Description
coverage_term_line_id	NUMBER	Coverage Term Line Id

Parameter	Data Type	Description
Coverage_term_name	VARCHAR2	Coverage Term Name
coverage_term_description	VARCHAR2	Coverage Term Description
coverage_level_line_id	NUMBER	Coverage Level Line Id
coverage_level	VARCHAR2	Coverage Level
coverage_level_code	VARCHAR2	Coverage Level Code
coverage_level_start_date	DATE	Coverage Level Start Date
coverage_level_End_date	DATE	Coverage Level End Date
coverage_level_id	NUMBER	Coverage Level Id
warranty_flag	VARCHAR2	Warranty Flag
eligible_for_entitlement	VARCHAR2	Eligible For Entitlement

Get Contracts

This is an over loaded API which returns contract information for different combination of Service, Extended Warranty or Warranty, Coverage Levels such as Party, Customer, Site, System, Item or Product and Business Processes. If the input parameter validate_flag is `Y', request date is checked against Date Effectivity and only those contract service lines eligible for entitlements are returned. The output table will be sorted in the order of ascending resolution time. The Sort_key accepts values 'RCN', 'RSN', or 'COVTYP_IMP' for sorting the output result based on reaction time, resolution time, or coverage importance level respectively.The output table of records returns coverage type, its importance level, service PO number, Service PO required flag, Contract Header Currency Code and Preventive Maintenance (PM) program Id and PM schedule exists flag.

Procedure Specification

```
PROCEDURE get_contracts
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_inp_rec              IN  get_contin_rec
```

```
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_ent_contracts        OUT NOCOPY get_contop_tbl);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_inp_rec	get_contin_rec	Yes	Input record. See the Data Structure Specification: get_contin_rec

get_contin_rec

Record Specification

```
TYPE get_contin_rec IS RECORD
(contract_number        OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,service_line_id        Number
,party_id               Number
,site_id                Number
,cust_acct_i            Number
,system_id              Number
,item_id                Number
,product_id             Number
,request_date           Date
,business_process_id    Number
,severity_id            Number
,time_zone_id           Number
```

```
,calc_resptime_flag     Varchar2(1)
,validate_flag          Varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
contract_number	VARCHAR2	No	Contract Number
service_line_id	NUMBER	Yes	Service Line Id. This input is required if input contract_number is also passed.
party_id	NUMBER	No	Party Id
site_id	NUMBER	No	Site Id
cust_acct_i	NUMBER	No	Customer Account Id
system_id	NUMBER	No	System Id
item_id	NUMBER	No	Item Id
product_id	NUMBER	No	Product Id
request_date	DATE	No	Request Date
business_process_id	NUMBER	Yes	Business Process Id - Required if calc_resptime_flag is Y.
severity_id	NUMBER	Yes	Severity Id - Required if calc_resptime_flag is Y.
time_zone_id	NUMBER	Yes	Time Zone Id - Required if calc_resptime_flag is Y.

Parameter	Data Type	Required	Description
calc_resptime_flag	VARCHAR2	No	Calculate Reaction and Resolution Time Flag Y/N. Valid values are 'Y' or, 'N'
validate_flag	VARCHAR2	No	Validate Flag Y/N. See API description for more information. Valid values are 'Y' or, 'N'. Default value is 'N'.
Sort_key	VARCHAR2	No	The Sort key used to sort the output table of records. This input is optional. The default sort_key used is 'RCN' for sorting based on resolution time.Sort_key accepts values 'RCN', 'RSN', or 'COVTYP_IMP' for sorting the output result based on reaction time, resolution time, or coverage importance level respectively. Default value is 'N'.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter

Parameter	Data Type	Description
x_ent_contracts	get_contop_tbl	Contract information. See the Data Structure Specification: get_contop_tbl

get_contop_tbl

Record Specification

```
TYPE get_contop_tbl IS TABLE OF get_contop_rec
INDEX BY BINARY_INTEGER;
TYPE get_contop_rec IS RECORD
(contract_id                Number
,contract_number            OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,contract_number_modifier   OKC_K_HEADERS_B.
CONTRACT_NUMBER_MODIFIER%TYPE
,sts_code                   OKC_K_HEADERS_B.STS_CODE%TYPE
,service_line_id            Number
,service_name               OKX_SYSTEM_ITEMS_V.NAME%TYPE
,service_description        OKX_SYSTEM_ITEMS_V.DESCRIPTION%TYPE
,coverage_term_line_id      Number
,coverage_term_name         OKC_K_LINES_V.NAME%TYPE
,coverage_term_description  OKC_K_LINES_V.ITEM_DESCRIPTION%TYPE
,service_start_date         Date
,service_end_date           Date
,warranty_flag              Varchar2(1)
,eligible_for_entitlement   Varchar2(1)
,exp_reaction_time          Date
,exp_resolution_time        Date
,status_code                Varchar2(1)
,status_text                Varchar2(1995)
,date_terminated            Date
,PM_Program_Id              VARCHAR2(40)
,PM_Schedule_Exists         VARCHAR2(450)
,HD_Currency_code           Varchar2(15)
,Service_PO_Number          VARCHAR2(450)
,Service_PO_Required_flag   VARCHAR2(1);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
contract_id	NUMBER	Contract Id
contract_number	VARCHAR2	Contract Number
contract_number_modifier	VARCHAR2	Contract Number Modifier
sts_code	VARCHAR2	Contract Status Code
service_line_id	NUMBER	Service Line Id
service_name	VARCHAR2	Service Name
service_description	VARCHAR2	Service Description
coverage_term_line_id	NUMBER	Coverage Term Line Id
coverage_term_name	VARCHAR2	Coverage Term Name
coverage_term_description	VARCHAR2	Coverage Term Description
service_start_date	NUMBER	Service Start Date
service_end_date	NUMBER	Service End Date
warranty_flag	VARCHAR2	Warranty Flag
eligible_for_entitlement	VARCHAR2	Eligible For Entitlement
exp_reaction_time	DATE	Expected Reaction Time
exp_resolution_time	DATE	Expected Resolution Time

Parameter	Data Type	Description
status_code	VARCHAR2	Status Code after returning reaction and/or resolution time S - Success E - Error U - Unexpected Error.
status_text	VARCHAR2	Status Text for the above Status Code.
date_terminated	DATE	The date terminated
PM_Program ID	NUMBER	Preventive Maintenance (PM) program ID
PM_Schedule_Exists	VARCHAR2	Schedule Exists flag for the PM Program ID
HD_Currency_Code	VARCHAR2	Contract Header currency code
Service_PO_Number	VARCHAR2	Service PO number
Service_PO_Required_Flag	VARCHAR2	Flag indicates if Service PO required.

Get Contracts

This is an over loaded API which returns contract information for different combination of Service, Extended Warranty or Warranty, Coverage Levels such as Party, Customer, Site, System, Item or Product and Business Processes. If the input parameter validate_flag is `Y', only those contract service lines eligible for entitlements are returned. The output table is sorted in the order of ascending resolution time or, Coverage type importance level based on the sort key. The output returns coverage type, coverage type importance level, and date terminated information.

Procedure Specification

```
PROCEDURE get_contracts
(p_api_version          IN  Number
```

```
,p_init_msg_list        IN  Varchar2
,p_inp_rec              IN  input_rec_ib
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_ent_contracts        OUT NOCOPY output_tbl_ib);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_inp_rec	input_rec_ib	Yes	Input record. See the Data Structure Specification: input_rec_ib

input_rec_ib

Record Specification

```
TYPE input_rec_ib IS RECORD
(contract_number           OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,contract_number_modifier OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
,service_line_id          Number
,party_id                 Number
,site_id                  Number
,cust_acct_id             Number
,system_id                Number
,item_id                  Number
,product_id               Number
,business_process_id      Number
```

```
,severity_id              Number
,time_zone_id             Number
,calc_resptime_flag       Varchar2(1)
,validate_flag            Varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
contract_number	VARCHAR2	No	Contract Number
contract_number_modifier	VARCHAR2	Yes	Contract Number Modifier. This input is required if input contract_number is also passed.
service_line_id	NUMBER	No	Service Line Id
party_id	NUMBER	No	Party Id
site_id	NUMBER	No	Site Id
cust_acct_id	NUMBER	No	Customer Account Id
system_id	NUMBER	No	System Id
item_id	NUMBER	No	Item Id
product_id	NUMBER	No	Product Id
business_process_id	NUMBER	Yes	Business Process Id - Required if calc_resptime_flag is Y.
severity_id	NUMBER	Yes	Severity Id - Required if calc_resptime_flag is Y.

Parameter	Data Type	Required	Description
time_zone_id	NUMBER	Yes	Time Zone Id - Required if calc_resptime_flag is Y.
calc_resptime_flag	VARCHAR2	No	Calculate Reaction and Resolution Time Flag Y/N. Valid values are 'Y' or 'N'. Default value is 'N'.
validate_flag	VARCHAR2	No	Validate Flag Y/N. Valid values are 'Y' or 'N'. Default value is 'N'. See the API description for more information.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_ent_contracts	output_tbl_ib	Contract information. See the Data Structure Specification: output_tbl_ib

output_tbl_ib

Record Specification

```
TYPE output_tbl_ib IS TABLE OF output_rec_ib
INDEX BY BINARY_INTEGER;
TYPE output_rec_ib IS RECORD
(contract_id                Number
,contract_number           OKC_K_HEADERS_B.CONTRACT_NUMBER%TYPE
```

```
,contract_number_modifier  OKC_K_HEADERS_B.
CONTRACT_NUMBER_MODIFIER%TYPE
,sts_code                  OKC_K_HEADERS_B.STS_CODE%TYPE
,service_line_id           Number
,service_name              OKX_SYSTEM_ITEMS_V.NAME%TYPE
,service_description       OKX_SYSTEM_ITEMS_V.DESCRIPTION%TYPE
,coverage_term_line_id     Number
,coverage_term_name        OKC_K_LINES_V.NAME%TYPE
,coverage_term_description OKC_K_LINES_V.ITEM_DESCRIPTION%TYPE
,service_start_date        Date
,service_end_date          Date
,warranty_flag             Varchar2(1)
,eligible_for_entitlement  Varchar2(1)
,exp_reaction_time         Date
,exp_resolution_time       Date
,status_code               Varchar2(1)
,status_text               Varchar2(1995)
,date_terminated           date);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
contract_id	NUMBER	Contract Id
contract_number	VARCHAR2	Contract Number
contract_number_modifier	VARCHAR2	Contract Number Modifier
sts_code	VARCHAR2	Contract Status Code
service_line_id	NUMBER	Service Line Id
service_name	VARCHAR2	Service Name
service_description	VARCHAR2	Service Description
coverage_term_line_id	NUMBER	Coverage Term Line Id

Parameter	Data Type	Description
coverage_term_name	VARCHAR2	Coverage Term Name
coverage_term_description	VARCHAR2	Coverage Term Description
Coverage_type_code	VARCHAR2	Coverage Type Code
Coverage_type_imp_level	NUMBER	Coverage type importance level
service_start_date	DATE	Service Start Date
service_end_date	DATE	Service End Date
warranty_flag	VARCHAR2	Warranty Flag
eligible_for_entitlement	VARCHAR2	Eligible For Entitlement
exp_reaction_time	DATE	Exp Reaction Time
exp_resolution_time	DATE	Exp Resolution Time
status_code	VARCHAR2	Status Code after returning reaction and/or resolution time. S - Success E - Error U - Unexpected Error.
status_text	VARCHAR2	Status Text for the above Status Code.
Date_terminated	DATE	Date Terminated

Get Coverage Levels

This API returns the Coverage Level such as Party, Customer, Site, System, Item and Product information for a Service, Extended Warranty or Warranty.

Procedure Specification

```
PROCEDURE get_coverage_levels
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_contract_line_id     IN  Number
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_covered_levels       OUT NOCOPY clvl_tbl_type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_contract_line_id	NUMBER	Yes	Line ID of Service, Extended Warranty or Warranty.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter

Parameter	Data Type	Description
x_covered_levels	clvl_tbl_type	Coverage Level information. See the Data Structure Specification: clvl_tbl_type

clvl_tbl_type

Record Specification

```
TYPE clvl_tbl_type IS TABLE OF clvl_rec_type
INDEX BY BINARY_INTEGER;
TYPE clvl_rec_type IS RECORD
(ROW_ID                 ROWID
,LINE_ID                OKC_K_LINES_B.ID%TYPE
,HEADER_ID              OKC_K_LINES_B.CHR_ID%TYPE
,PARENT_LINE_ID         OKC_K_LINES_B.CLE_ID%TYPE
,LINE_LEVEL             Varchar2(150)
,CP_ID                  Number
,CP_NAME                Varchar2(240)
,INV_ITEM_ID            Number
,ITEM_NAME              Varchar2(240)
,SITE_ID                Number
,SITE_NAME              Varchar2(240)
,SYSTEM_ID              Number
,SYSTEM_NAME            Varchar2(240)
,CUSTOMER_ID            Number
,CUSTOMER_NAME          Varchar2(240)
,PARTY_ID               Number
,PARTY_NAME             Varchar2(500)
,QUANTITY               Number
,LIST_PRICE             Number
,PRICE_NEGOTIATED       OKC_K_LINES_B.PRICE_NEGOTIATED%TYPE
,LINE_NAME              Varchar2(150)
,DEFAULT_AMCV_FLAG      Varchar2(1)
,DEFAULT_QTY            Number
,DEFAULT_UOM            Varchar2(25)
,DEFAULT_DURATION       Number
,DEFAULT_PERIOD         Varchar2(25)
```

```
,MINIMUM_QTY            Number
,MINIMUM_UOM            Varchar2(25)
,MINIMUM_DURATION       Number
,MINIMUM_PERIOD         Varchar2(25)
,FIXED_QTY              Number
,FIXED_UOM              Varchar2(25)
,FIXED_DURATION         Number
,FIXED_PERIOD           Varchar2(25)
,LEVEL_FLAG             Varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
row_id	ROWID	Row Id
line_id	NUMBER	Line Id
header_id	NUMBER	Header Id
parent_line_id	NUMBER	Parent Line Id
line_level	VARCHAR2	Covered Level
cp_id	NUMBER	Customer Product Id
cp_name	VARCHAR2	Customer Product Name
inv_item_id	NUMBER	Inventory Item Id
item_name	VARCHAR2	Item Name
site_id	NUMBER	Site Id
site_name	VARCHAR2	Site Name
system_id	NUMBER	System Id
system_name	VARCHAR2	System Name

Parameter	Data Type	Description
customer_id	NUMBER	Customer Id
customer_name	VARCHAR2	Customer Name
party_id	NUMBER	Party Id
party_name	VARCHAR2	Party Name
quantity	NUMBER	Quantity
list_price	NUMBER	List Price
price_negotiated	NUMBER	Price Negotiated
line_name	VARCHAR2	Line Name
default_amcv_flag	VARCHAR2	Default Amcv Flag
default_qty	NUMBER	Default Quantity
default_uom	VARCHAR2	Default UOM
default_duration	NUMBER	Default Duration
default_period	VARCHAR2	Default Period
minimum_qty	NUMBER	Minimum Quantity
minimum_uom	VARCHAR2	Minimum UOM
minimum_duration	NUMBER	Minimum Duration
minimum_period	VARCHAR2	Minimum Period
fixed_qty	NUMBER	Fixed Quantity
fixed_uom	VARCHAR2	Fixed UOM
fixed_duration	NUMBER	Fixed Duration

Parameter	Data Type	Description
fixed_period	VARCHAR2	Fixed Period
level_flag	VARCHAR2	Level Flag

Get Contacts

This API returns the Contact information for a contract or a line.

Procedure Specification

```
PROCEDURE get_contacts
(p_api_version          IN  Number
,p_init_msg_list        IN  Varchar2
,p_contract_id          IN  Number
,p_contract_line_id     IN  Number
,x_return_status        OUT NOCOPY Varchar2
,x_msg_count            OUT NOCOPY Number
,x_msg_data             OUT NOCOPY Varchar2
,x_ent_contacts         OUT NOCOPY ent_contact_tbl);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter

Parameter	Data Type	Required	Description and Validations
p_contract_id	NUMBER	No	Contract Header ID. Either p_contract_id, or p_contract_line_id is required.
p_contract_line_id	NUMBER	No	Contract Line ID of Service, Extended Warranty or Warranty. Either p_contract_id, or p_contract_line_id is required.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_ent_contacts	ent_contact_tbl	Contact information. See the Data Structure Specification ent_contact_tbl

ent_contact_tbl

Record Specification

```
TYPE ent_contact_tbl IS TABLE OF ent_contact_rec
INDEX BY BINARY_INTEGER;
TYPE ent_contact_rec IS RECORD
(contract_id            Number
,contract_line_id      Number
,contact_id            Number
,contact_name          Varchar2(50)
,contact_role_id       Number
```

```
,contact_role_code     Varchar2(30)
,contact_role_name     Varchar2(80));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
contract_id	NUMBER	Contract Id
contract_line_id	NUMBER	Contract Line Id
contact_id	NUMBER	Contact Id
contact_name	VARCHAR2	Contact Name
contact_role_id	NUMBER	Contact Role Id
contact_role_code	VARCHAR2	Contact Role Code
contact_role_name	VARCHAR2	Contact Role Name

Get Preferred Engineers

This API returns the details of Preferred Engineers for a Service, Extended Warranty or Warranty.

Procedure Specification

```
PROCEDURE get_preferred_engineers
(p_api_version            IN  Number
,p_init_msg_list          IN  Varchar2
,p_contract_line_id       IN  Number
,P_business_process_id    IN  NUMBER default NULL
,P_request_date           IN  DATE default sysdate
,x_return_status          OUT NOCOPY Varchar2
,x_msg_count              OUT NOCOPY Number
,x_msg_data               OUT NOCOPY Varchar2
,x_prf_engineers          OUT NOCOPY prfeng_tbl_type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_contract_line_id	NUMBER	Yes	Contract Line ID of a of Service, Extended Warranty or Warranty.
P_business_process_id	NUMBER	No	Business Process ID.
P_request_date	DATE	No	Request date. The default is sysdate.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_prf_engineers	prfeng_tbl_type	Contact information. See the Data Structure Specification prfeng_tbl_type

prfeng_tbl_type

Record Specification

```
TYPE prfeng_tbl_type IS TABLE OF prfeng_rec_type

INDEX BY BINARY_INTEGER;

TYPE prfeng_rec_type IS RECORD

( business_process_id number
,engineer_id       Number
,resource_type    Varchar2(30)
,primary_flag     varchar2(1)
,resource_class   varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Business_process_id	NUMBER	Business process id
engineer_id	NUMBER	Engineer Id
resource_type	VARCHAR2	Resource Type
Primary_flag	VARCHAR2	Primary Flag
Resource_class	VARCHAR2	Resource Class

Get Coverage Type

This API returns the coverage type and importance level information for a contract line id.

Procedure Specification

```
PROCEDURE Get_Coverage_Type
(P_API_Version       IN  NUMBER
,P_Init_Msg_List     IN  VARCHAR2
,P_Contract_Line_Id  IN  NUMBER
```

```
,X_Return_Status     out nocopy VARCHAR2
,X_Msg_Count         out nocopy NUMBER
,X_Msg_Data          out nocopy VARCHAR2
,X_Coverage_Type     out nocopy CovType_Rec_Type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_contract_line_id	NUMBER	Yes	Contract Line ID of a of Service, Extended Warranty or Warranty

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description and Validations
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_coverage_type	Covtype_rec_type	Coverage type information. See the Data Structure Specification: covtype_rec_type

covtype_rec_type

Record Specification

```
TYPE CovType_Rec_Type IS RECORD
(Code                Oks_Cov_Types_B.Code%TYPE
,Meaning             Oks_Cov_Types_TL.Meaning%TYPE
,Importance_Level    Oks_Cov_Types_B.Importance_Level%TYPE);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Code	VARCHAR2	Coverage Type Code
Meaning	VARCHAR2	Coverage Type Meaning
Importance_level	NUMBER	Coverage type Importance Level

Get Cov Txn Groups

This API returns the business process line level information in the coverage based on the business process setup done for a given input of contract line id.

Procedure Specification

```
PROCEDURE Get_cov_txn_groups
(p_api_version    IN  Number
,p_init_msg_list  IN  Varchar2
,p_inp_rec_bp     IN  inp_rec_bp
,x_return_status  out nocopy Varchar2
,x_msg_count      out nocopy Number
,x_msg_data       out nocopy Varchar2
,x_cov_txn_grp_linesout nocopy output_tbl_bp);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_inp_rec_bp	Inp_rec_bp	Yes	Contract line information. See the Data Structure Specification: inp_rec_bp

inp_rec_bp

Record Specification

```
TYPE INP_REC_BP IS RECORD
(contract_line_id      NUMBER
,check_bp_def          VARCHAR2(1)
,sr_enabled            VARCHAR2(1)
,dr_enabled            VARCHAR2(1)
,fs_enabled            VARCHAR2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
contract_line_id	NUMBER	Yes	Contract line id
Check_bp_def	VARCHAR2	No	Check Business process definition

Parameter	Data Type	Required	Description
Sr_enabled	VARCHAR2	No	Flag to check if business process is setup for service request
Dr_enabled	VARCHAR2	No	Flag to check if business process is setup for depot repair
Dr_enabled	VARCHAR2	No	Flag to check if business process is setup for field service

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_cov_txn_grp_lines	Output_tbl_bp	Coverage Business process line level information. See the Data Structure Specification: output_tbl_bp

output_tbl_bp

Record Specification

```
TYPE output_tbl_bp IS TABLE OF output_rec_bp INDEX BY BINARY_INTEGER;
TYPE output_rec_bp  IS RECORD
(cov_txn_grp_line_id    NUMBER
,bp_id                  number
,start_date             date
,end_date               date);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Cov_txn_group_line_id	NUMBER	Business process line id in the coverage
Bp_id	NUMBER	Business process id
Start_date	DATE	Business process line start date
End date	DATE	Business process line end date

Get Txn Billing Types

This API returns the service activity billing type line level information and labor bill rate level information in the coverage based on the coverage business process line id (transaction group line id).

Procedure Specification

```
PROCEDURE Get_txn_billing_types
(p_api_version            IN Number
,p_init_msg_list          IN Varchar2
,p_cov_txngrp_line_id     IN Number
,p_return_bill_rates_YN   IN Varchar2
,x_return_status          out nocopy Varchar2
,x_msg_count              out nocopy Number
,x_msg_data               out nocopy Varchar2
,x_txn_bill_types         out nocopy output_tbl_bt
,x_txn_bill_rates         out nocopy output_tbl_br);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_cov_txngrp_line_id	NUMBER	Yes	Coverage transaction group line id
P_return_bill_rates_yn	VARCHAR2	Yes	Flag to indicate if labor bill rates to be returned as output. Valid values are 'Y' or, 'N'.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_txn_bill_types	Output_tbl_bt	Coverage service activity billing type line level information. See the Data Structure Specification: output_tbl_bt
X_txn_bill_rates	Output_tbl_br	Coverage labor bill rate line level information. See the Data Structure Specification: output_tbl_br

output_tbl_bt

Record Specification

```
TYPE output_tbl_bt IS TABLE OF output_rec_bt INDEX BY BINARY_INTEGER;

TYPE output_rec_bt IS RECORD
(Txn_BT_line_id         NUMBER
,txn_bill_type_id       Number
,Covered_upto_amount    Number
,percent_covered        Number);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
txn_bt_line_id	NUMBER	Coverage Service activity billing type line id
Txn_bill_type_id	NUMBER	Service activity billing type id
Covered_upto_amount	NUMBER	The amount covered by the Coverage Service activity billing type line id
Percent_covered	NUMBER	The percent covered by the Coverage Service activity billing type line id

output_tbl_br

Record Specification

```
TYPE output_tbl_br IS TABLE OF output_rec_br INDEX BY BINARY_INTEGER;

TYPE output_rec_br IS RECORD
(BT_line_id                  NUMBER
,Br_line_id                  NUMBER
,Br_schedule_id              NUMBER
,bill_rate                   VARCHAR2(30)
,flat_rate                   NUMBER
,uom                         VARCHAR2(30)
```

```
,percent_over_list_price      NUMBER
,start_hour                   NUMBER
,start_minute                 NUMBER
,end_hour                     NUMBER
,end_minute                   NUMBER
,monday_flag                  VARCHAR2(1)
,tuesday_flag                 VARCHAR2(1)
,wednesday_flag               VARCHAR2(1)
,thursday_flag               VARCHAR2(1)
,friday_flag                  VARCHAR2(1)
,saturday_flag                VARCHAR2(1)
,sunday_flag                  VARCHAR2(1)
,labor_item_org_id            number
,labor_item_id                number
,holiday_yn                   VARCHAR2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
bt_line_id	NUMBER	Coverage Service activity billing type line id
Br_line_id	NUMBER	Coverage labor bill rate line id
Br_schedule_id	NUMBER	Coverage labor bill rate schedule id
Bill_rate	VARCHAR2	Labor bill rate code
Flat rate	NUMBER	Flat rate for labor bill rate code
UOM	VARCHAR2	UOM
Percent_over_list_price	NUMBER	Percent over list price
Start_hour	NUMBER	Labor bill rate schedule start hour

Parameter	Data Type	Description
Start_minute	NUMBER	Labor bill rate schedule start minute
End_hour	NUMBER	Labor bill rate schedule end hour
End_minute	NUMBER	Labor bill rate schedule end minute
Monday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Monday
Tuesday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Tuesday
Wednesday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Wednesday
Thursday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Thursday
Friday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Friday
Saturday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Saturday
Sunday_flag	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for Sunday
Labor_item_org_id	NUMBER	Inventory Labor item organization id for the Labor bill rate schedule
Labor_item_id	NUMBER	Inventory Labor item id for the Labor bill rate schedule

Parameter	Data Type	Description
Holiday_YN	VARCHAR2	Flag indicating if Labor bill rate schedule is applicable for holidays

Get Contracts Expiration

This API returns the expiration details, that is, contract end date, contract grace period and duration, for a given contract id.

Procedure Specification

```
PROCEDURE Get_Contracts_Expiration
(p_api_version              IN  Number
,p_init_msg_list            IN  Varchar2
,p_contract_id              IN  Number
,x_return_status            out nocopy Varchar2
,x_msg_count                out nocopy Number
,x_msg_data                 out nocopy Varchar2
,x_contract_end_date        out nocopy date
,x_Contract_Grace_Duration  out nocopy number
,x_Contract_Grace_Period    out nocopy VARCHAR2);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter

Parameter	Data Type	Required	Description and Validations
p_contract_id	NUMBER	Yes	Contract Id

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_contract_end_date	DATE	Contract End date
x_contract_grace_duration	NUMBER	Contract grace duration
x_contract_grace_period	VARCHAR2	Contract grace period

Validate Contract line

This API returns if the contract line id is valid for the given input of covered level table of records and the business process id.

Procedure Specification

```
PROCEDURE VALIDATE_CONTRACT_LINE
(p_api_version              IN  NUMBER
,p_init_msg_list            IN  VARCHAR2
,p_contract_line_id         IN  NUMBER
,p_busiproc_id              IN  NUMBER
,p_request_date             IN  DATE
,p_covlevel_tbl_in          IN  covlevel_tbl_type
,p_verify_combination       IN  VARCHAR2
,x_return_status            out nocopy Varchar2
,x_msg_count                out nocopy Number
,x_msg_data                 out nocopy Varchar2
```

```
,x_covlevel_tbl_out         OUT NOCOPY  covlevel_tbl_type
,x_combination_valid        OUT NOCOPY VARCHAR2);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_contract_line_id	NUMBER	Yes	Contract line Id
P_busiproc_id	NUMBER	Yes	Business process ID
P_request_date	DATE	No	Request date.Default is sysdate.
P_covlevel_tbl_in	Covlevel_tbl_type	Yes	Covered level code and id. See the Data Structure Specification: covlevel_tbl_type
P_verify_combination	VARCHAR2	No	Default 'N'. If 'Y', then the procedure checks if all the covered level records passed as input in p_covlevel_tbl_in are valid and p_busiproc_id is valid for the p_contract_line_id

covlevel_tbl_type

Record Specification

```
TYPE covlevel_tbl_type IS TABLE OF COVLEVEL_REC INDEX BY BINARY_INTEGER
;

TYPE COVLEVEL_REC IS RECORD
(covlevel_code                VARCHAR2(50),
covlevel_id                  NUMBER,
inv_org_id                   NUMBER,
covered_yn                   VARCHAR2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Covlevel_code	VARCHAR2	Covered level code. This input is required. The covered level codes are: For install base customer product; 'OKX_CUSTPROD', for inventory item; 'OKX_COVITEM', for install base system;'OKX_COVSYST', for customer account;'OKX_CUSTACCT',for customer party site;'OKX_PARTYSITE',for customer party;'OKX_PARTY'
Covlevel_id	NUMBER	Covered level Id corresponding to covlevel code. This input is required. For example, The covered level id would be an install base item instance id, if covered_level_code = 'OKX_CUSTPROD'
Inv_org_id	NUMBER	Inventory_organization_id. This Input is required if input covlevel_code is passed as 'OKX_COVITEM'.

Parameter	Data Type	Description
Covered_yn	VARCHAR2	Not Used in the input record. This is used while returning the output. Same data structure is used both for input and output.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
x_covlevel_tbl_out	Covlevel_tbl_type	Covered level code and id passed as input is returned with covered_yn information. See the Data Structure Specification: covlevel_tbl_type
X_combination_valid	VARCHAR2	Returns 'Y' or, 'N'. If 'Y', means all of the records in the input p_covlvl_tbl_in and the p_busiproc_id is valid.

covlevel_tbl_type

Record Specification

```
TYPE covlevel_tbl_type IS TABLE OF COVLEVEL_REC INDEX BY BINARY_INTEGER
;

TYPE COVLEVEL_REC IS RECORD
(covlevel_code                VARCHAR2(50),
covlevel_id                  NUMBER,
inv_org_id                   NUMBER,
covered_yn                   VARCHAR2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Covlevel_code	VARCHAR2	Covered level code. Input is returned as it is.
Covlevel_id	NUMBER	Covered level Id corresponding to covlevel code. Input is returned as it is.
Inv_org_id	NUMBER	Inventory_organization_id for covlevel_code = 'OKX_COVITEM'. Input is returned as it is.
Covered_yn	VARCHAR2	returns 'Y' or, 'N'. 'Y' means for the given contract_line_id this particular covered level record is valid.

Package OKS_CON_COVERAGE_PUB

The Apply Contract Coverage APIs consist of the following procedures:

- Apply Contract Coverage, page 22-59
- Get BP Pricelist, page 22-63

Apply Contract Coverage

This API returns the discounted amount for a Service or Extended Warranty based on the Discount Amount and Percentage Covered, specified in the Service Contract coverage terms for a specific business process. Instead of passing txn_group_id as an input, user must pass the combination of business_process_id and request_date. The txn_group_id is retained for backward compatibility purpose only.

Procedure Specification

```
PROCEDURE apply_contract_coverage
(p_api_version            IN Number
,p_init_msg_list          IN Varchar2
```

```
,p_est_amt_tbl            IN ser_tbl_type
,x_return_status          OUT NOCOPY Varchar2
,x_msg_count              OUT NOCOPY Number
,x_msg_data               OUT NOCOPY Varchar2
,x_est_discounted_amt_tbl OUT NOCOPY cov_tbl_type);
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
p_est_amt_tbl	ser_tbl_type	Yes	Input table of records. See the Data Structure Specification: ser_tbl_type

ser_tbl_type

Input table for apply_contract_coverage.

Record Specification

```
TYPE ser_tbl_type IS TABLE OF ser_rec_type
INDEX BY BINARY_INTEGER;
TYPE ser_rec_type IS RECORD
(charges_line_number  Number
,estimate_detail_id  Number
,contract_line_id    Number
,txn_group_id        Number
,billing_type_id     Number
,charge_amount       Number);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
charges_line_number	NUMBER	Charges Line Number.This input is Optional.
estimate_detail_id	NUMBER	Estimate Detail Id.This input is Optional.
contract_line_id	NUMBER	Contract Line Id.This input is Required.
txn_group_id	NUMBER	Txn Group Id.This is kept for backward compatibility. Instead, please pass input Business process id
Business_process_id	NUMBER	Business process id.This input is required.
Request date	DATE	Request date.This input is Optional.Default is sysdate.
billing_type_id	NUMBER	Billing Type Id.This input is required.
charge_amount	NUMBER	Charge Amount.This input is required.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter

Parameter	Data Type	Description
x_est_discounted_acmt_tb1	cov_tbl_type	Output table of records. See the Data Structure Specification: cov_tbl_type

cov_tbl_type

Record Specification

```
TYPE cov_tbl_type IS TABLE OF cov_rec_type
INDEX BY BINARY_INTEGER;
TYPE cov_rec_type IS RECORD
(charges_line_number  Number
,estimate_detail_id  Number
,contract_line_id    Number
,txn_group_id        Number
,billing_type_id     Number
,discounted_amount   Number
,status              Varchar2(1));
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
charges_line_number	NUMBER	Charges Line Number
estimate_detail_id	NUMBER	Estimate Detail Id
contract_line_id	NUMBER	Contract Line Id
txn_group_id	NUMBER	Txn Group Id
Business_process_id	NUMBER	Business process id
Request date	DATE	Request date
billing_type_id	NUMBER	Billing Type Id

Parameter	Data Type	Description
discounted_amount	NUMBER	Discounted Amount
status	VARCHAR2	Status of returning records. T - Discount defined in coverage. F - Discount not defined in coverage

Get BP Pricelist

This API returns the price list and discount at the coverage business process level and the price list at the contract header level for a contract_lien_id, business_process_id, and request date.

Procedure Specification

```
PROCEDURE get_bp_pricelist
(p_api_version            IN  Number
,p_init_msg_list          IN  Varchar2
,p_Contract_line_id       IN NUMBER
,p_business_process_id    IN NUMBER
,p_request_date           IN DATE
,x_return_status          OUT NOCOPY Varchar2
,x_msg_count              OUT NOCOPY Number
,x_msg_data               OUT NOCOPY Varchar2
,x_pricing_tbl            OUT NOCOPY PRICING_TBL_TYPE );
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	Standard IN Parameter
p_init_msg_list	VARCHAR2	Yes	Standard IN Parameter
P_contract_line_id	NUMBER	Yes	Contract line Id
P_business_process_id	NUMBER	Yes	Business process ID
P_request_date	DATE	No	Default is sysdate.

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	Standard OUT Parameter
x_msg_count	NUMBER	Standard OUT Parameter
x_msg_data	VARCHAR2	Standard OUT Parameter
X_pricing_tbl	Pricing_tbl_type	Price list information. See the Data Structure Specification: pricing_tbl_type

pricing_tbl_type

Record Specification

```
TYPE pricing_tbl_type IS TABLE OF pricing_rec_type INDEX BY
BINARY_INTEGER;

TYPE pricing_rec_type IS RECORD

(contract_line_id                 Number,

business_process_id              NUMBER,

BP_Price_list_id                 NUMBER,

BP_Discount_id                   NUMBER,

BP_start_date                    DATE,
```

```
BP_end_date                     DATE,
Contract_Price_list_Id          NUMBER );
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
contract_line_id	NUMBER	Contract Line Id
Business_process_id	NUMBER	Business process id
BP_price_list_id	NUMBER	Coverage Business process level price list id
BP_Discount_id	NUMBER	Coverage Business process level discount price list id
BP_start_date	DATE	Coverage Business process line level start date
BP_end_date	DATE	Coverage Business process line level end date
Contract_Price_list_id	NUMBER	Contract header level price list id

Package OKS_OMINT_PUB

The OKS_OMINT_PUB consist of the following procedures:

- Get Duration, page 22-65
- Is Service Available, page 22-68
- Available Services, page 22-71
- OKS Available Services, page 22-73

Get Duration

This API is to calculate Service Duration by entering the Start_Date and End_Date (Optional). You can also choose to coterminate all the service programs that you order

so that the program ends at the same time as other service programs for the products associated with the same customer or system.

Procedure Specification

```
PROCEDURE Get_Duration
(
P_Api_Version          IN        Number,
P_init_msg_list        IN        Varchar2 Default OKC_API.G_FALSE,
X_msg_Count            OUT       Number,
X_msg_Data             OUT       Varchar2,
X_Return_Status        OUT       Varchar2,
P_customer_id          IN        Number,
P_system_id            IN        Number,
P_Service_Duration     IN        Number,
P_service_period       IN        Varchar2,
P_coterm_checked_yn    IN        Varchar2 Default OKC_API.G_FALSE,
P_start_date           IN        Date,
P_end_date             IN        Date,
X_service_duration     OUT       Number,
X_service_period       OUT       Varchar2,
X_new_end_date         OUT       Date
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	IN/OUT	Required	Description and Validations
p_api_version	NUMBER	IN	Yes	-
p_init_msg_list	VARCHAR2	IN	Yes	-
p_customer_id	NUMBER	IN	Yes (if system id is Null)	Customer Id
p_system_id	NUMBER	IN		System Id

Parameter	Data Type	IN/OUT	Required	Description and Validations
p_coterm_checked_yn	VARCHAR2	IN	No	Coterminate Flag Y/N
p_service_duration	NUMBER	IN	No	Duration of the Service
p_service_period	VARCHAR2	IN	No	Period of the Service
p_start_date	DATE	IN	Yes	Start Date of The Service
p_end_date	DATE	IN	No	End Date of the Service

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	fnd_api.g_ret_sts_success, fnd_api.g_ret_sts_error, fnd_api.g_ret_sts_unexp_error
x_msg_count	NUMBER	Message count in the message list.
x_msg_data	VARCHAR2	Message
x_service_duration	VARCHAR2	-
x_new_end_date	DATE	-
x_service_period	VARCHAR2	-

Profiles To Be Set

- OKS Minimum Service Duration
- OKS Minimum Service Period.

Validations

- API will check the minimum duration profile and Day_UOM, if it is not set it will error out.
- Will check if service duration is greater than or equal to minimum duration.
- If Cotermination flag is set, It will pull system or customer coterminate date and accordingly sets the service duration.
- The Coterminate date for a customer can be set in Cotermination set up form.
- Either P_end_date or (p_service_duration,p_service_period) is required if Coterminate flag not set.
- Service duration is calculated based on co-termination flag of customer/system co-termination dates. System Co Terminate Date gets the priority over customer co-termination date.

Is Service Available

This API can be used to check whether the given service is available for a product or for a customer.

Procedure Specification

```
PROCEDURE Is_service_available
(
P_Api_Version          IN  Number,
P_init_msg_list        IN  Varchar2 Default OKC_API.G_FALSE,
X_msg_Count            OUT Number,
X_msg_Data             OUT Varchar2,
X_Return_Status        OUT Varchar2,
p_check_service_rec    IN  CHECK_SERVICE_REC_TYPE,
X_Available_YN         OUT Varchar2
)
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	-
p_init_msg_list	VARCHAR2	Yes	default fnd_api.g_false
p_check_service_rec	CHECK_SERVICE_REC_TYPE	Yes	See the Data Structure Specification: check_service_rec_type

check_service_rec_type

Record Specification

```
TYPE CHECK_SERVICE_REC_TYPE IS RECORD
(
product_item_id           Number
,service_item_id          Number
,customer_id              Number
,customer_product_id      Number
,product_revision         Varchar2(3)
request_date              Date
)
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
product_item_id	NUMBER	Yes (If Customer id Is null)	The inventory item id of the product
service_item_id	NUMBER	Yes	Inventory Item ID of the Service
customer_id	NUMBER	Yes (If Product item id Is Null)	The Customer Id

Parameter	Data Type	Required	Description
customer_product_id	NUMBER	No	The Customer product Id
product_revision	VARCHAR2	No	The revision of the product
request_date	DATE	No	Default: SYSDATE, Otherwise Ordering date

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	fnd_api.g_ret_sts_success, fnd_api.g_ret_sts_error, fnd_api.g_ret_sts_unexp_error
x_msg_count	NUMBER	Message count in the message list.
x_msg_data	VARCHAR2	Message
x_Available_yn	VARCHAR2	Available Y/N

Functionality

- Returns Available as 'N' if that service item is not defined in Service Availability form and the customer product has service order allowed flag set to 'N'.
- Returns Available as 'Y' if that service item is not defined in Service Availability form and the customer product has service order allowed flag set to 'Y'.
- If Service item, Product item id and Customer id are passed then the procedure checks if the service is available for both the customer and the product item.
- It checks if request date of the service item available for the product item or customer falls within the effectivity of all the three parameters service item, product item and customer id.

Validations

- Will error out if Customer Id and Product item id are Null.
- The Product item id passed should be of a serviceable Item.

> **Note:** OKS Service Availability Form allows or restricts access to specific service programs for specific customers and products/revisions. It can be further delimited by entering starting and ending availability dates.

Available Services

This API returns list of services available to ORDER for a customer or product.

Procedure Specification

```
PROCEDURE Available_Services
(
P_Api_Version             IN      Number,
P_init_msg_list           IN      Varchar2 Default OKC_API.G_FALSE,
X_msg_Count               OUT     Number,
X_msg_Data                OUT     Varchar2,
X_Return_Status           OUT     Varchar2,
p_avail_service_rec       IN      AVAIL_SERVICE_REC_TYPE,
X_Orderable_Service_tbl   OUT     order_service_tbl_type
)
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	-

Parameter	Data Type	Required	Description and Validations
p_init_msg_list	VARCHAR2	Yes	default fnd_api.g_false
p_avail_service_rec_rec	avail_services_rec_type	Yes	See the Data Structure Specification: avail_service_rec_type

avail_service_rec_type

Record Specification

```
TYPE AVAIL_SERVICE_REC_TYPE  Is RECORD
(
product_item_id       Number
,customer_id          Number
,product_revision     Varchar2(3)
,request_date         Date
)
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
product_item_id	NUMBER	Yes (If Customer id Is null)	The inventory item id of the product
customer_id	NUMBER	Yes (If Product item id Is null)	The Customer Id
product_revision	VARCHAR2	No	The revision of the product
request_date	DATE	No	Default: SYSDATE, Otherwise Ordering date

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	fnd_api.g_ret_sts_success, fnd_api.g_ret_sts_error, fnd_api.g_ret_sts_unexp_error
x_msg_count	NUMBER	Message count in the message list.
x_msg_data	VARCHAR2	Message
x_orderable_services_rec	order_service_tbl_type	Service Item information. See the Data Structure Specification: order_service_tbl_type

order_service_tbl_type

Record Specification

```
TYPE inp_rec_type IS RECORD
(Service_item_id OUT NUMBER);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Service_item_id	NUMBER	Service Item

OKS Available Services

This API returns the list of services with Name, Description and the Coverage template associated to the Service available to ORDER for a customer or product.

Procedure Specification

```
PROCEDURE Available_Services
(
P_Api_Version            IN     Number,
```

```
P_init_msg_list          IN     Varchar2 Default OKC_API.G_FALSE,
X_msg_Count              OUT    Number,
X_msg_Data               OUT    Varchar2,
X_Return_Status          OUT    Varchar2,
p_avail_service_rec      IN     AVAIL_SERVICE_REC_TYPE,
X_Orderable_Service_tbl  OUT    oks_order_service_tbl_type
)
```

Current Version

1.0

Parameter Descriptions

The following table describes the IN parameters associated with this API.

Parameter	Data Type	Required	Description and Validations
p_api_version	NUMBER	Yes	-
p_init_msg_list	VARCHAR2	Yes	default fnd_api.g_false
p_avail_service_rec_rec	avail_service_rec_type	Yes	See the Data Structure Specification: avail_service_rec_type.

avail_service_rec_type

Record Specification

```
TYPE AVAIL_SERVICE_REC_TYPE  Is RECORD
(
product_item_id       Number
,customer_id          Number
,product_revision     Varchar2(3)
,request_date         Date
)
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Required	Description
product_item_id	NUMBER	Yes (If Customer id Is null)	The inventory item id of the product
customer_id	NUMBER	Yes (If Product item id Is null)	The Customer Id
product_revision	VARCHAR2	No	The revision of the product
request_date	DATE	No	Default: SYSDATE, Otherwise Ordering date

The following table describes the OUT parameters associated with this API.

Parameter	Data Type	Description
x_return_status	VARCHAR2	fnd_api.g_ret_sts_success, fnd_api.g_ret_sts_error, fnd_api.g_ret_sts_unexp_error
x_msg_count	NUMBER	Message count in the message list.
x_msg_data	VARCHAR2	Message
x_orderable_service_tbl	oks_order_service_tbl_type	Service Item information. See the Data Structure Specification: oks_order_service_tbl_type

oks_order_service_tbl_type

Record Specification

```
TYPE order_service_rec_type IS RECORD
(Service_item_id       NUMBER,
Name                  VARCHAR2,
Description           VARCHAR2,
```

```
Coverage_template_id  NUMBER);
```

Parameter Descriptions

The following table describes the parameters associated with this data structure.

Parameter	Data Type	Description
Service_item_id	NUMBER	Service Item
Name	VARCHAR2	Name Of service
Description	VARCHAR2	Description Of service
Coverage_template_id	NUMBER	Coverage Associated with Service

Index

L

M

O

P

R

S

T

U

W