Oracle® Manufacturing APIs and Open Interfaces Manual

Volume 1

Release 11i

February 2001

Part No. A88838-01

Oracle© Manufacturing APIs and Open Interfaces Manual, Release 11i

Part No. A88838-01

Contributing Authors: Jill Arehart, Rachel Haas, Elizabeth Looney, Tom Myers, Susan Saperstein, Amy Sonczalla

Contributors: David Reitan

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

Contents

4 Oracle Cost Management Open Interfaces

5 Engineering Change Order Business Object Interface

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

8 Oracle Purchasing Open Interfaces

9 Oracle Quality Open Interfaces

Send Us Your Comments

Oracle Manufacturing APIs and Open Interfaces Manual, Release 11i

Part No. A88838-01

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

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

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

- E-mail - your_product@us.oracle.com
- FAX - telephone number.   Attn: Oracle Manufacturing APIs and Open Interfaces
- Postal service: 3op

  Oracle Corporation
  Oracle Manufacturing APIs and Open Interfaces Documentation
  500 Oracle Parkway, 3op
  Redwood Shores, CA 94065
  United States

If you would like a reply, please give your name, address, and telephone number below.

____

____

____

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

Preface

This *Oracle® Manufacturing APIs and Open Interfaces Manual, Release 11i* contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems.

This preface explains how you should use this manual, and defines the notational conventions you need to understand.

> **Note:** This documentation includes open interfaces found in Oracle Manufacturing Release 11i. If you are using an earlier version of the software, please consult your support representative or the Product Update Notes for more specific information about your release.

About This Manual

This manual contains information about importing/exporting information using Oracle Applications open interfaces. This manual includes the following chapters:

- Chapter 1 gives you an overview of Manufacturing integration tools and explains how to use these tools to integrate Oracle Manufacturing products with one another and with non-Oracle systems.
- Chapter 2 contains information about Oracle ASCP and Oracle Global ATP Server.
- Chapter 3 contains information about Oracle Bills of Material open interfaces.
- Chapter 4 contains information about Oracle Cost Management open interfaces.
- Chapter 5 contains information about Oracle Engineering open interfaces.

- Chapter 6 contains information about Oracle Inventory open interfaces.
- Chapter 7 contains information about Oracle Master Scheduling/MRP and Oracle Supply Chain Planning open interfaces.
- Chapter 8 contains information about Oracle Purchasing or Oracle Public Sector Purchasing open interfaces.
- Chapter 9 contains information about Oracle Quality open interfaces.
- Chapter 10 contains information about Oracle Work in Process open interfaces.

Audience for This Manual

This manual provides you information needed to integrate with other Oracle Manufacturing and Distribution applications and your other systems. This manual is intended for the use of the team implementing Oracle Manufacturing applications. In order to effectively implement, this team should include all levels of individuals including but not limited to:

- Project Leaders
- Systems Analysts
- Department managers
- Application Programmers
- System Programmers
- System Managers
- Database Administrators

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle Applications tables are interrelated and any change you make using Oracle Applications can update several 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.

Consequently, we STRONGLY RECOMMEND that you never use SQL*Plus or any other tool to modify Oracle Applications data unless otherwise instructed, or when working within an open interface table as described in this manual.

Other Information Sources

Here are some other ways you can increase your knowledge and understanding of Oracle Manufacturing and Distribution applications.

Online Documentation

All Oracle Applications documentation is available online on CD-ROM, except for technical reference manuals. There are two online formats, HyperText Markup Language (HTML) and Adobe Acrobat (PDF).

All user's guides are available in HTML, Acrobat, and paper. Technical reference manuals are available in paper only. Other documentation is available in Acrobat and paper.

The *content* of the documentation does not differ from format to format. There may be slight differences due to publication standards, but such differences do not affect content. For example, page numbers and screen shots are not included in HTML.

The HTML documentation is available from all Oracle Applications windows. Each window is programmed to start your web browser and open a specific, context-sensitive section. Once any section of the HTML documentation is open, you can navigate freely throughout all Oracle Applications documentation. The HTML documentation also ships with Oracle Information Navigator (if your national language supports this tool), which enables you to search for words and phrases throughout the documentation set.

Related User's Guides

Oracle Manufacturing and Distribution applications share business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user's guides when you are integrating your systems.

If you do not have the hardcopy versions of these manuals, you can read them online using the Applications Library icon or Help menu command.

Oracle Applications User's Guide This guide explains how to enter data, query, run reports, and navigate using the graphical user interface (GUI) available with this release of Oracle Applications products. This guide also includes information on setting user profiles, as well as running and reviewing reports and concurrent processes.

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

Oracle Applications Demonstration User's Guide This guide documents the functional storyline and product flows for Global Computers, a fictional manufacturer of personal computers products and services. As well as including product overviews, the book contains detailed discussions and examples across each of the major product flows. Tables, illustrations, and charts summarize key flows and data elements.

Reference Manuals

Oracle Automotive Implementation Manual This manual describes the setup and implementation of the Oracle Applications used for the Oracle Automotive solution.

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

Oracle Project Manufacturing Implementation Manual This manual describes the setup steps and implementation for Oracle Project Manufacturing.

Oracle Self-Service Web Applications Implementation Manual This manual describes the setup steps for Oracle Self-Service Web Applications and the Web Applications dictionary.

Installation and System Administration

Oracle Alert User's Guide This guide explains how to define periodic and event alerts to monitor the status of your Oracle Applications data.

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

Multiple Organizations in Oracle Applications If you use the Oracle Applications Multiple Organization Support feature to use multiple sets of books for one product installation, this guide describes all you need to know about setting up and using the product with this feature.

Oracle Applications Implementation Wizard User's Guide If you are implementing more than one Oracle product, you can use the Oracle Applications Implementation Wizard to coordinate your setup activities. This guide describes how to use the wizard.

Oracle Applications Developer's Guide This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards*. It also provides information to help you build your custom Developer/2000 forms so that they integrate with Oracle Applications.

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

Oracle Applications Installation Manual for Windows Clients This guide provides information you need to successfully install Oracle Financials, Oracle Public Sector Financials, Oracle Manufacturing, or Oracle Human Resources in your specific hardware and operating system software environment.

Oracle Applications Product Update Notes If you are upgrading your Oracle Applications, refer to the product update notes appropriate to your update and product(s) to see summaries of new features as well as changes to database objects, profile options and seed data added for each new release.

Oracle Applications Upgrade Preparation Manual This guide explains how to prepare your Oracle Applications products for an upgrade. It also contains information on completing the upgrade procedure for each product. Refer to this manual and the *Oracle Applications Installation Manual* when you plan to upgrade your products.

Oracle Applications System Administrator's Guide This manual provides planning and reference information for the System Administrator.

Other Sources

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

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

Support From on-site support to central support, our team of experienced professionals provides the help and information you need to keep the product working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle8 server, and your hardware and software environment.

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support, and office automation, as well as Oracle Applications, an integrated suite of more than 70 software modules for financial management, supply chain management, manufacturing, project systems, human resources and sales and service management.

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

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and applications products, along with related consulting, education, and support services, in over 140 countries around the world.

Thank You

Thank you for using Oracle Manufacturing applications and this implementation manual.

We value your comments and feedback. At the end of this manual is a Reader's Comment Form you can use to explain what you like or dislike about Oracle Manufacturing applications or this implementation manual. Mail your comments to the following address or call us directly at (650) 506-7000.

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

Or, send electronic mail to **appsdoc@us.oracle.com**.

1

Integrating Your Systems

This chapter gives you an overview of Manufacturing integration tools and explains how to use these tools to integrate Oracle Manufacturing products with one another and with your existing non–Oracle systems.

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

- Overview of Oracle Manufacturing APIs and Open Interfaces on page 1-2

Overview of Oracle Manufacturing APIs and Open Interfaces

Oracle Manufacturing products provide a number of open interfaces so you can link them 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 Manufacturing 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 Manufacturing product APIs and open interfaces provide you with the features you need to support the following basic business needs:

- Connect to data collection devices. This lets you collect material movement transactions such as receipts, issues, quality data, movements, completions, and shipments. This 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 via electronic commerce.
- Control processing of inbound data imported from outside Oracle applications.
- Validate imported data to ensure integrity of Oracle Manufacturing products.
- Review, update, and resubmit imported data that failed validation.
- Export data from Oracle Manufacturing products

Oracle Manufacturing Interfaces

Open Interface Architectures

Oracle Manufacturing products have 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 Manufacturing 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 Manufacturing products. They are not *interfaces* as much as *open integration*.

For example, note flexfield 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 which provide import and export capabilities and allow 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

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

Key	
Data Flow Direction	*Inbound* means into Oracle Manufacturing; *Outbound* means out from Oracle Manufacturing
Iface Man	The interface is documented in detail in the *Oracle Manufacturing and Distribution Open Interfaces Manual*
TRM	The tables, views, or modules are described in the product's Technical Reference Manual

Table 1–1 Oracle Manufacturing Interfaces/APIs

Interface/API Name	Data Flow Direction	Table, View, Process, or Procedure	Iface Man	TRM	Table, View, Module Name, or Procedure Name
Oracle Inventory					
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

Table 1–1 Oracle Manufacturing Interfaces/APIs

Interface/API Name	Data Flow Direction	Table, View, Process, or Procedure	Iface Man	TRM	Table, View, Module Name, or Procedure Name
Material	Inbound	Table	Yes	Yes	MTL_TRANSACTIONS_INTERFACE MTL_SERIAL_NUMBERS_INTERFACE MTL_TRANSACTION_LOTS_INTERFACE
Oracle Engineering / Oracle Bills of Material					
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
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)
Oracle Master Scheduling/MRP and Oracle Supply Chain Planning					
Forecast Interface	Inbound	Table	Yes	Yes	MRP_FORECAST_INTERFACE

Table 1–1 Oracle Manufacturing Interfaces/APIs

Interface/API Name	Data Flow Direction	Table, View, Process, or Procedure	Iface Man	TRM	Table, View, Module Name, or Procedure Name
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
Oracle Purchasing					
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
Receiving	Inbound	Table	Yes	Yes	RCV_HEADERS_INTERFACE RCV_TRANSACTIONS_INTERFACE
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
Receiving	Inbound	Table	Yes	Yes	RCV_HEADERS_INTERFACE RCV_TRANSACTIONS_INTERFACE

Table 1–1 Oracle Manufacturing Interfaces/APIs

Interface/API Name	Data Flow Direction	Table, View, Process, or Procedure	Iface Man	TRM	Table, View, Module Name, or Procedure Name
Oracle Quality					
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
Oracle Work in Process					
Moves	Inbound	Table	Yes	Yes	WIP_MOVE_TXN_INTERFACE
Resource	Inbound	Table	Yes	Yes	WIP_COST_TXN_INTERFACE
Work Order Interface	Inbound	Table	Yes	Yes	WIP_JOB_SCHEDULE_INTERFACE WIP_JOB_DTLS_INTERFACE

Inbound Open Interface Model

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

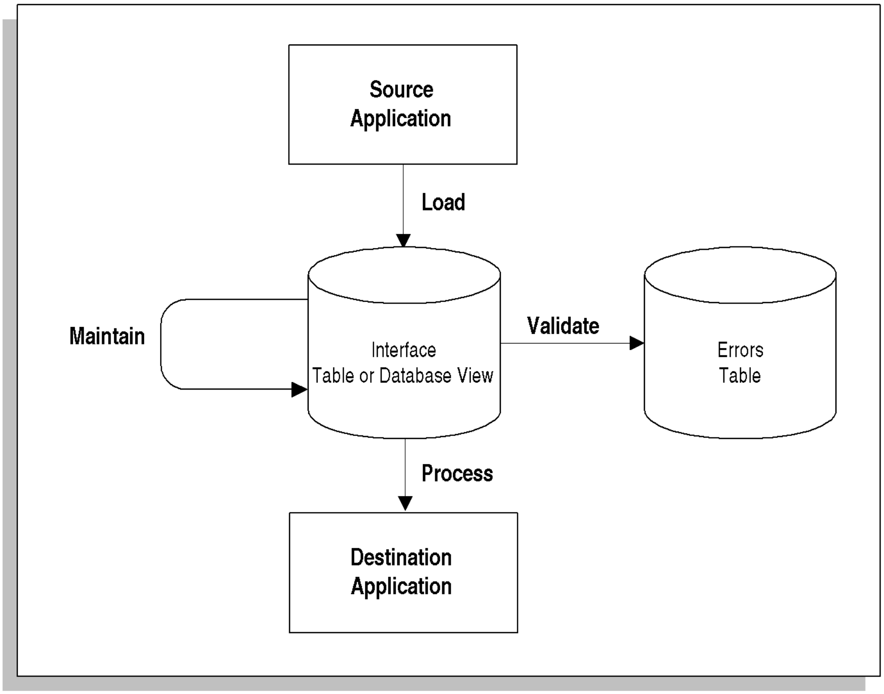
Discussion of Inbound Interfaces Only

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:

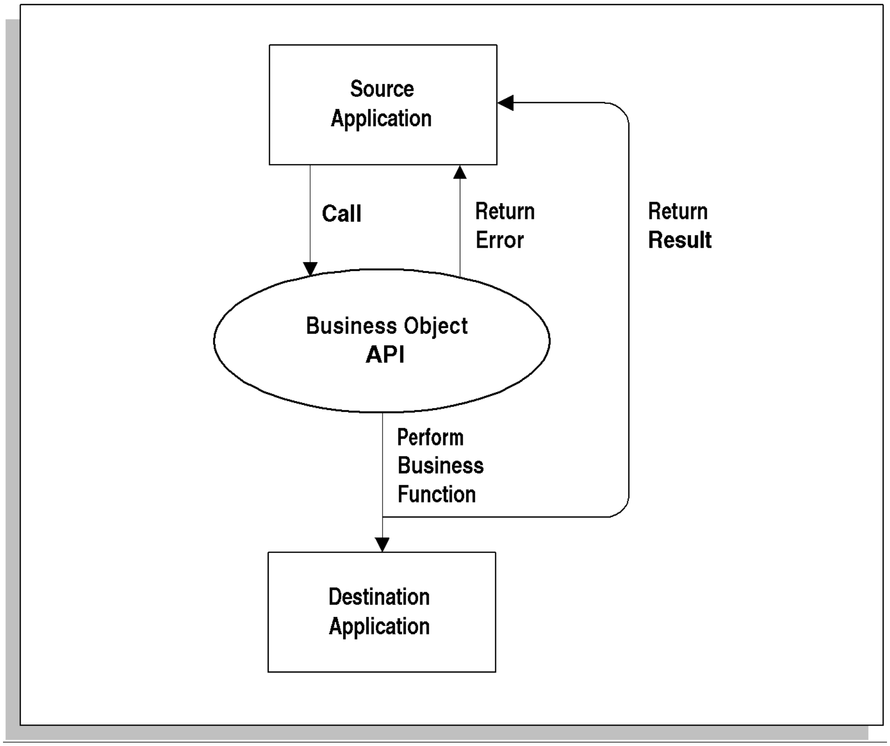
Figure 1–1 Open Interface Diagram



Open Application Programmatic Interface (API) Diagram

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

Figure 1–2 Open Application Programmatic Interface (API) Diagram



Components of an Open Interface

There are a number of components that are generally common to all open interfaces. These components are described below. However, all open interfaces do not include every component, and in some cases the component may be implemented slightly differently than described below.

Source Application

You obtain data from a source application to pass on to a destination application for further processing and/or storage. Typically the data has completed processing in the source application before being passed.

Oracle Manufacturing 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/or storage.

Oracle Manufacturing 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 Manufacturing product. The various types of interface columns, with examples from the Oracle Work in Process Move Transaction Interface, are listed below:

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 Manufacturing product. For example, transaction quantity is one attribute of a move transaction.

Required Columns Required columns store the minimum information needed by the Oracle Manufacturing product to successfully process the interface row. For example, 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 Derived columns are created by the destination product from information in the required columns. For example, on a move transaction, the primary unit of measure is derived from the assembly being moved.

Optional Columns Optional columns are not necessarily required by Oracle Manufacturing products but can be used for additional value-added functionality. For example, for move transactions the reason code is not required, but can optionally be used 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 allows 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, which requires you to 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, Order Import 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 Manufacturing source products available for selection and use by destination applications.

Oracle Manufacturing 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 data can be selected by the destination application.

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 Manufacturing 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 Manufacturing product, you would likely use a tool such as Pro*C or PL/SQL since 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 Manufacturing destination products use to insure 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, since Oracle Manufacturing 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 Manufacturing destination application.

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

For example, several validation tasks are performed by the move transaction validation program 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

The move transaction validation program also derives columns from required columns such as WIP_ENTITY_ID from the job name and PROD_LINE_ID from the line code.

When an Oracle Manufacturing product is the source product, the destination application should provide the validate function.

Process Function

The process function is a set of programs that processes the data from the interface table into the Oracle Manufacturing destination product. The specific processing performed varies by application. For open transaction interfaces, the processing generally includes recording transaction history, updating inventory and order balances, and charging costs.

Interfaces typically let you control both the frequency and the number of validated rows that the processing programs attempt to process. Upon successful completion of processing, the process function should delete the processed row from the interface table.

On occasion, the process function may need to insert rows into the errors table. For example, if the Oracle Work in Process Open Move Transaction Interface processing function encounters problems such as lack of grants, it updates the interface table with an error status and inserts an error in the errors table.

When an Oracle Manufacturing product is the source, the destination application should provide the process function.

Maintain Function

The maintain function is generally accomplished from a window within an Oracle Manufacturing product. Most of these windows allow you to query, update, and resubmit interface records that have validation. You can generally use these windows to query unprocessed or unvalidated rows and check their current status.

For example, if invalid information from a bar code device is inserted into the Oracle Work in Process Open Move Transaction interface table, the load validation function catches the error. You must either fix or delete the problem record using the Pending Move Transactions window. Corrected rows can be resubmitted for processing. Deleting problem data, then subsequently correcting it at the source, ensures that you do not have duplicate data when the information is reinserted.

In the case where there is no formal maintain function, you can use SQL*Plus to query and update the errored interface table rows.

When an Oracle Manufacturing product is the source application, the destination application should provide the maintain function.

2

Oracle ASCP and Oracle Global ATP Server Open Interfaces

This section explains how to use the ODS Load API and how it functions in Oracle ASCP and Oracle Global ATP Server. Topics included are:

- ODS Load API Features on page 2-2
- Functional Overview on page 2-7
- Setting Up the ODS Load API on page 2-7

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		
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		
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		
INCLUDE_DISCRETE_WIP_DEMAND	IN	NUMBER	x		
INCLUDE_REP_WIP_DEMAND	IN	NUMBER	x		

Parameter	Usage	Type	Required	Derived	Optional
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		
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
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

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		

Parameter	Usage	Type	Required	Derived	Optional
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	
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
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	

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		

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
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		

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		
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

Parameter	Usage	Type	Required	Derived	Optional
RANGE2_LOW	IN	NUMBER			x
RANGE2_HIGH	IN	NUMBER			x
RANGE3_LOW	IN	NUMBER			x
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	
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		

Parameter	Usage	Type	Required	Derived	Optional
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
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	
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	
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
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		

Parameter	Usage	Type	Required	Derived	Optional
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		
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
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

Parameter	Usage	Type	Required	Derived	Optional
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		
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		
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
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	

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

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		

Parameter	Usage	Type	Required	Derived	Optional
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
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
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	

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

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		

Parameter	Usage	Type	Required	Derived	Optional
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	
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	
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		
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	
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
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		

Parameter	Usage	Type	Required	Derived	Optional
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
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
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	

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
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
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

Parameter	Usage	Type	Required	Derived	Optional
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
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		
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	
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
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		
OWNING_DEPARTMENT_ID	IN	NUMBER			x
CAPACITY_UNITS	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
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
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
LINE_DISABLE_DATE	IN	DATE			x

Parameter	Usage	Type	Required	Derived	Optional
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		
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		

Parameter	Usage	Type	Required	Derived	Optional
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
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
CONSUME_FORECAST	IN	NUMBER			x
UPDATE_TYPE	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
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		
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

Parameter	Usage	Type	Required	Derived	Optional
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	
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

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
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	

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

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
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	
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	

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	
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	
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		

Parameter	Usage	Type	Required	Derived	Optional
BILL_SEQUENCE_ID	IN	NUMBER	x		
ROUTING_SEQUENCE_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	
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		
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		
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	
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		
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

Parameter	Usage	Type	Required	Derived	Optional
PLAN_DATE_DEFAULT_TYPE	IN	NUMBER			x
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
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

Parameter	Usage	Type	Required	Derived	Optional
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	
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	
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
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	

Parameter	Usage	Type	Required	Derived	Optional
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
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	
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		

Parameter	Usage	Type	Required	Derived	Optional
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
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
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

Parameter	Usage	Type	Required	Derived	Optional
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	
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		
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	
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

MANAGER_CONTACT

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		

Parameter	Usage	Type	Required	Derived	Optional
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
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	
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
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	

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		

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		
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	
ATTRIBUTE_CATEGORY	IN	VARCHAR2(30)			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
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

Parameter	Usage	Type	Required	Derived	Optional
ASSIGNED_UNITS	IN	NUMBER	x		
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	
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	

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		

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		
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		

Parameter	Usage	Type	Required	Derived	Optional
OPERATION_TYPE	IN	NUMBER			x
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
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		

Parameter	Usage	Type	Required	Derived	Optional
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
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		
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

Parameter	Usage	Type	Required	Derived	Optional
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	
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
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

Parameter	Usage	Type	Required	Derived	Optional
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		

Parameter	Usage	Type	Required	Derived	Optional
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	
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

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	

Parameter	Usage	Type	Required	Derived	Optional
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
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		
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		

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	

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

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	
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	

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

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
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	

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
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	
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

Parameter	Usage	Type	Required	Derived	Optional
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	
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
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

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		
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
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

Parameter	Usage	Type	Required	Derived	Optional
OLD_ORDER_PLACEMENT_DATE	IN	DATE			x
RESCHEDULE_DAYS	IN	NUMBER			x
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
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

Parameter	Usage	Type	Required	Derived	Optional
IMPLEMENT_AS	IN	NUMBER			x
RELEASE_STATUS	IN	NUMBER			x
LOAD_TYPE	IN	NUMBER			x
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
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

Parameter	Usage	Type	Required	Derived	Optional
IMPLEMENT_TASK_ID	IN	NUMBER(15)			x
IMPLEMENT_SCHEDULE_GROUP_ID	IN	NUMBER			x
IMPLEMENT_BUILD_SEQUENCE	IN	NUMBER			x
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
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	

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		
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
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

Parameter	Usage	Type	Required	Derived	Optional
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

Parameter	Usage	Type	Required	Derived	Optional
FIXED_ORDER_QUANTITY	IN	NUMBER			x
FIXED_LOT_MULTIPLIER	IN	NUMBER			x
MINIMUM_ORDER_QUANTITY	IN	NUMBER			x
MAXIMUM_ORDER_QUANTITY	IN	NUMBER			x
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
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		

Parameter	Usage	Type	Required	Derived	Optional
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
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
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

Parameter	Usage	Type	Required	Derived	Optional
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
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
ORGANIZATION_CODE	IN	VARCHAR2(7)			x
EFFECTIVITY_CONTROL	IN	NUMBER	x		
ACCEPTABLE_EARLY_DELIVERY	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
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	
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
RESOURCE_COST	IN	NUMBER			x

Parameter	Usage	Type	Required	Derived	Optional
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

Parameter	Usage	Type	Required	Derived	Optional
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
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	
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
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
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		

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
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		
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	

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	
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		

Parameter	Usage	Type	Required	Derived	Optional
COMMENTS	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	
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		

Parameter	Usage	Type	Required	Derived	Optional
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	
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	

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_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

3

Bills of Material Business Object Interface

Topics covered in this chapter include:

Overview

The Bills of Material Business Object Interface allows 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, 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 Business Object Interface can process all types of bills, including model, option class, planning, standard and product family. 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.

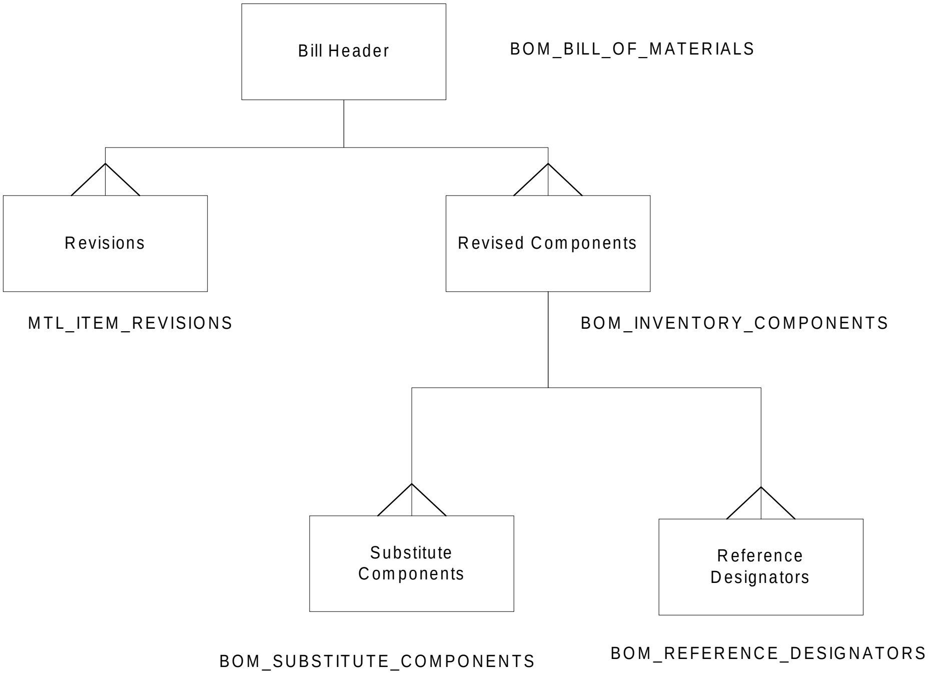
This document describes the basic business needs, major features, business object architecture and components for the Insert, Update and Delete features for the Bills of Material Business Object Interface.

Features

- Creating, Updating, and Deleting Bills of Material Information. The Bills of Material Business Object Interface lets 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 that record as a delete.
- Bills of Material Business Object Data Encapsulation. When you import bills of material from an external system it is not required to provide system specific information. All the Bills of material business object interface requires is business specific data that is required to define a bill.
- Synchronous Processing of information within a Bill. The Bills of Material Business Object Interface will process the information within a bill synchronously. Following the business hierarchy, it will process bill header information before components.
- Asynchronous Processing of Bills of Material. You can process multiple bills simultaneously using the Bills of Material Business object interface.
- Detailed and Translatable Error Messages. If your import fails, the Bills of Material Business Object Interface will report detailed and translatable error messages. The error message will identify the severity of the error and scope with which the error affects other record that you have tried to import in this business object.

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 like the header item name, organization in which the item exists, alternate designator if bill is an alternate and the revision.

Revisions: Revisions is a direct child of the header. You can 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: Components are a direct child of the header entity. They cannot exist without a parent item. To identify a component, you must specify the identity of the parent by giving the assembly item number, organization in which it exists, alternate designator if the component is being added to an alternate bill, effective date of the component, the component item number, and its operation sequence number.

Substitute Component: The substitute components entity is a direct child of the components entity. It cannot exist without a component. To identify a substitute component, you must identify the parent by giving organization in which it exists, the assembly item number, its effectivity date, alternate if the component belongs on an alternate bill, the component item number, its operation sequence number, the substitute component number.

Reference Designator: The reference designator entity is a direct child of the components entity. It cannot exist without a component. To identify a substitute component, you must identify the parent by giving organization in which it exists, the assembly item number, its effectivity date, alternate if the component belongs on an alternate bill, the component item number, its operation sequence number, the substitute component number.

The Business Object as it exists in the database

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
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.

In the database, the entity relationships are established through primary and foreign keys.

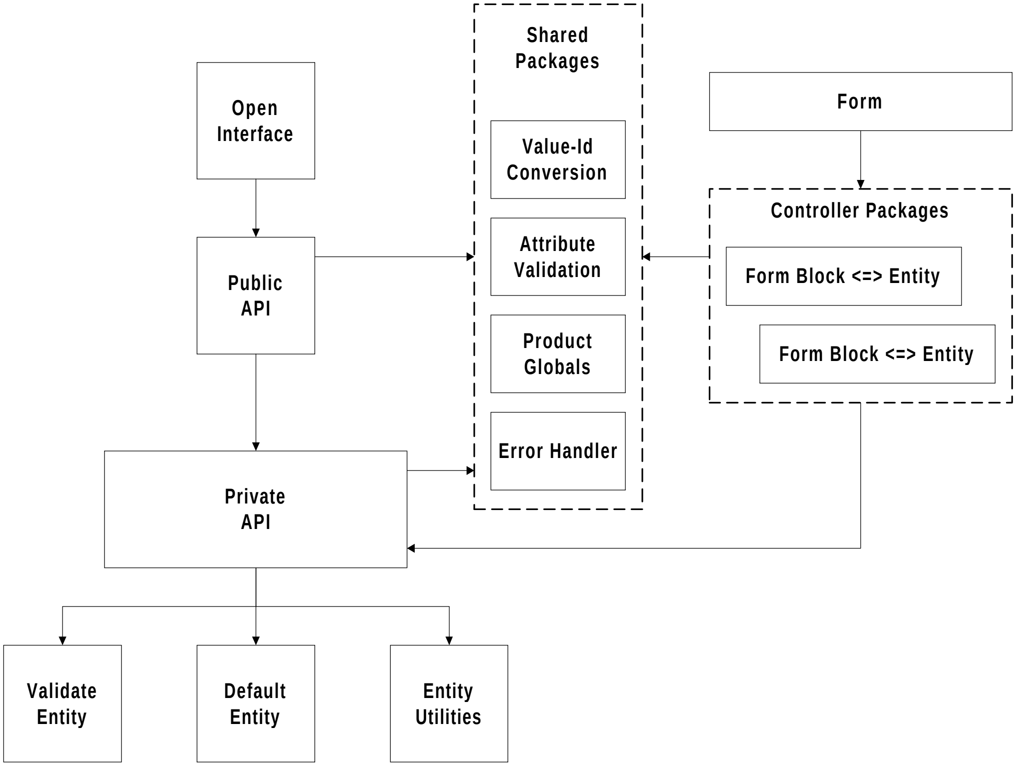
Business Logic and Business Object Rules Business logic helps model the business. It includes all constraints and considerations that the business needs to maintain and process data successfully. This implies that certain rules must be imposed on incoming data (business objects) to ensure its validity within the context of the business.

Resulting Design Considerations The program must be able to work with both the user's view of the business object, a hierarchy of entities, and the database's view, a group of inter-related tables. It must impose data entry and data manipulation rules in order to maintain the integrity of the business object. In other words, it should impart the user just enough flexibility to manipulate whole or sections of business objects, while never violating either the business object rules or the database table inter-relationships.

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.

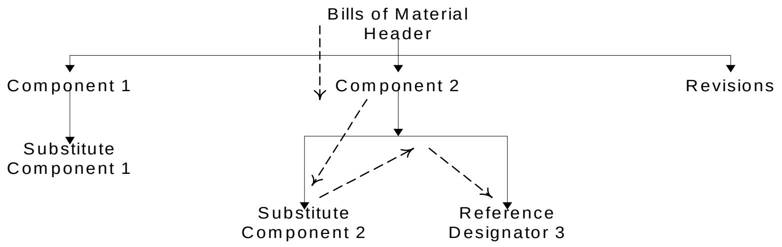


See the APIs section for an explanation of each of the framework components.

Business object interface Design

Business Object Traversal Strategy

Requirements The Business object architecture is a hierarchy of entities. It can also be viewed as an inverted tree, where the Bills of Material Header entity is the parent, and each of it's 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 it's own. Here is an example:



Objectives:

The Bills of Material business object is a closely tied object. If a user does not have access to a assembly item type item, say, then the user does not have access to any entity records associated with that bill. Hence, certain assembly item characteristics and errors will affect it’s children, and, sometimes the entire business object records. So, the program must traverse the tree in an efficient manner in order to achieve the following:

- Cut down on the amount of processing required. Make use of specific business objects facts that were established while processing other business object records.
- Provide intelligent error handling. Sometimes, an error in a parent record implies that there is an error in all of its children, so, if the program is unable to create a assembly item header record, it will not create any components on the Bill. The program must be able to infer this early, to avoid processing these components and any of its children.

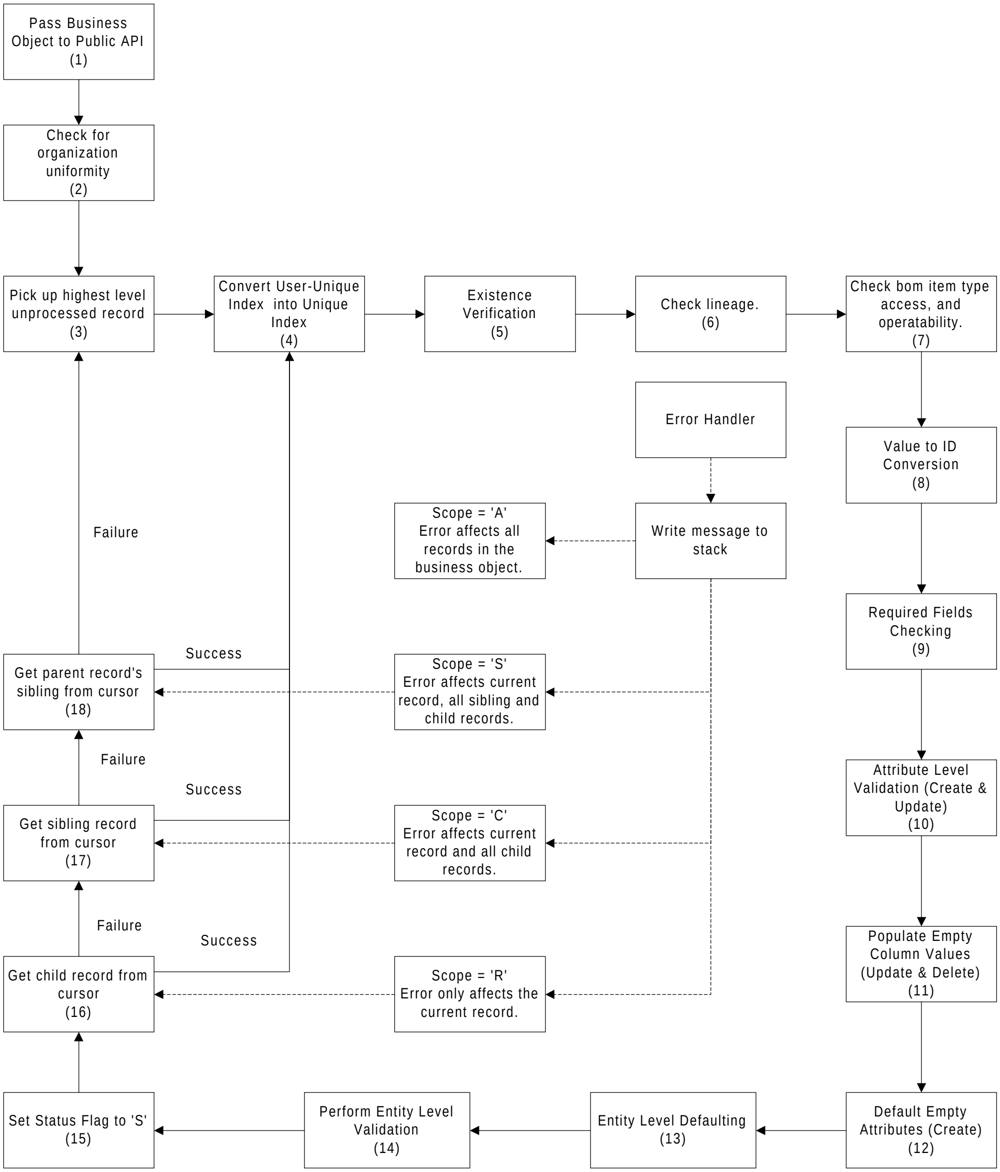
Solution:

A DEPTH-FIRST algorithm will be used. Process the children of a parent node in sequence. If a child is, in turn, the parent of more records, then process it’s branch by first processing the child (which is the parent here), and it’s children. This continues until the whole tree has been traversed. The example above will be processed:

1. First process the Bills of Material header node.
2. Then, traverse all of it’s children. Identify the first branch which is the one belonging to Component 1. Traverse this branch in the following manner:
    1. First process the parent node, Component 1.
    2. Process all it’s children, Substitute Component 1.
    3. Then process Revised Component 2 and process substitute component 2 and reference designator 3 … and so on for any other components.

Data Flow for CREATES, UPDATES and DELETES of Business object interface records into Production tables

Detailed Business object interface Design


Pass Business Object to Public API (1)
Check for organization uniformity (2)
Pick up highest level unprocessed record (3)
Convert User-Unique Index into Unique Index (4)
Existence Verification (5)
Check lineage. (6)
Check bom item type access, and operatability. (7)
Value to ID Conversion (8)
Required Fields Checking (9)
Attribute Level Validation (Create & Update) (10)
Populate Empty Column Values (Update & Delete) (11)
Default Empty Attributes (Create) (12)
Entity Level Defaulting (13)
Perform Entity Level Validation (14)
Set Status Flag to 'S' (15)
Get child record from cursor (16)
Get sibling record from cursor (17)
Get parent record's sibling from cursor (18)
Error Handler
Write message to stack
Scope = 'A' Error affects all records in the business object.
Scope = 'S' Error affects current record, all sibling and child records.
Scope = 'C' Error affects current record and all child records.
Scope = 'R' Error only affects the current record.
Failure
Success
Failure
Success
Failure
Success

This section provides a detailed description of all the Process Flow steps. It justifies the sequencing of the events by pointing out their relevance in the Process Flow. It also describes the errors that may occur, and how they are handled by the program.

Private Data Structures

There are some data structures that are private to the program, which drive the import.

OLD Records All entities have OLD records. These are used for UPDATEs and DELETEs. The production table record that is being updated or deleted is queried up into the associated OLD record structure. These records have the same columns as the associated production table records. These OLD records are for reference only and should not be written to once they have been queried up.

Unexposed Records All entities have unexposed records. The columns are different from the exposed columns. Users do not have access to the values stored in the unexposed records. These records are used to store the ID's or any system specific information that is derived during the processing of the business object entity.

SYSTEM_INFORMATION Record There are two kinds of information this record will store:

- System-specific information, such as profile values, program id, etc.
- Information gathered while processing a parent or sibling, which can be used when processing other records. It reduces the number of database hits in retrieving the same information for several records, so the program can skip some checks.

Information in this record may be reset per record, entity, or business object based whether it is useful and valid when processing another record, entity or business object.

Field	Type	Description
Entity	VARCHAR2(30)	Entity currently being processing. Reset per entity.
Org_Id	NUMBER	Organization the business object exists in. Reset per business object.
User_Id	NUMBER	Derived from the user environment. Reset per session.
Login_Id	NUMBER	Derived from the user environment. Reset per session.
Prog_Appid	NUMBER	Derived from the user environment. Reset per session.

Prog_Id	NUMBER	Derived from the user environment. Reset per session.
STD_Item_Access	BOOLEAN	Does the user have access to Standard Items ? Check profile ENG: Standard Item Change Order Access. Reset per business object.
MDL_Item_Access	BOOLEAN	Does the user have access to Model Items ? Check profile ENG: Model Item Change Order Access. Reset per business object.
PLN_Item_Access	BOOLEAN	Does the user have access to Planning Items ? Check profile ENG: Planning Item Change Order Access. Reset per business object.
Bill_Sequence_Id	NUMBER	Bill_Sequence_Id value generated for new alternate bills created for assembly items. Reset per assembly item.
Current_Revision	VARCHAR2(3)	Holds the most recently implemented revision for a assembly item. Reset per assembly item.

Overall Import Description

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 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 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 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 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. 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 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 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 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-

Columns Exposed to User

The following columns are exposed to the user.

Bills of Material Exposed Columns

Name	Associated Column Name	Type	Comments
Assembly_Item_Name	Assembly_Item_Id	VARCHAR2(81)	User-Unique Index
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Alternate_BOM_Code	Alternate_BOM_Designator	VARCHAR2(10)	User-Unique Index
Common_Assembly_Name	Common_Assembly_Item_Id	VARCHAR2(81)	
Common_Organization_Code	Common_Organization_Id	VARCHAR2(3)	
Assembly_Type	Assembly_Type	NUMBER	
Transaction_Type	-N/A-	VARCHAR2(30)	Required
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)	

Bills of Material Components Exposed Columns

Name	Associated Column Name	Type	Comments
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Assembly_Item_Name	Assembly_Item_Id	VARCHAR2(81)	User-Unique Index
Alternate_Bom_Code	Alternate_bom_desginator	VARCHAR2(10)	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
Transaction_Type	-N/A-	VARCHAR2(30)	Required
Disable_Date	Disable_Date		
New_Effectivity_Date	Effectivity_Date	DATE	
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	
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
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)	
From_End_Item_Unit_Number	From_End_Item_Unit_Number	VARCHAR2(30)	
To_End_Item_Unit_Number	To_End_Item_Unit_Number	VARCHAR2(30)	

Item Revision Exposed Columns

Name	Associated Column Name	Type	Comments
Organization_Code	Organization_id	VARCHAR2(3)	User-Unique Index
Assembly_Item_Name	Inventory_Item_Id	VARCHAR2(81)	User-Unique Index
Start_Effective_Date	Effectivity_Date	VARCHAR2(3)	User-Unique Index
Revision	Revision	VARCHAR2(15)	User-Unique Index
Transaction_Type	-N/A-	VARCHAR2(30)	Required
Revision_Comment	Description	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)	
Attribute14	Attribute14	VARCHAR2(150)	

Attribute15	Attribute15	VARCHAR2(150)	

Reference Designator Exposed Columns

Name	Associated Column Name	Type	Comments
Organization_Code	Organization_Code	VARCHAR2(3)	User-Unique Index
Assembly_Item_Name	Assembly_Item_Id	VARCHAR2(81)	User-Unique Index
Start_Effective_Date	Effectivity_Date	VARCHAR2(3)	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
Reference_Designator_Name	Component_Reference_Designator	VARCHAR2(15)	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)	

Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Substitute Component Exposed Columns

Name	**Name**	**Type**	**Comments**
Organization_Code	Organization_Id	VARCHAR2(3)	User-Unique Index
Assembly_Item_Name	Assembly_Item_Id	VARCHAR2(81)	User-Unique Index
Start_Effective_Date	Effectivity_Date	VARCHAR2(3)	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
Transaction_Type	-N/A	VARCHAR2(30)	Required
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)	

Import Mechanics

Record Structures

This section lists the data structures with the *exposed* columns. They are declared in the Public API specification package. The user needs to populate these data structures to import data.

BOM Header

```
TYPE Assembly_Item_Rec_Type IS RECORD
(   Organization_Code            VARCHAR2(3)    := FND_API.G_MISS_CHAR
,   Assembly_Item_Name           VARCHAR2(81)   := FND_API.G_MISS_CHAR
,   Alternate_Bom_Designator     VARCHAR2(10)   := FND_API.G_MISS_CHAR
,   Change_Description           VARCHAR2(240)  := FND_API.G_MISS_CHAR
,   Commn_Assembly_Item_Name     VARCHAR2(81)   := FND_API.G_MISS_CHAR
,   Common_Organization_Code     VARCHAR2(3)    := FND_API.G_MISS_CHAR
,   Assembly_Type    NUMBER      := FND_API.G_MISS_NUM
,   Attribute_category           VARCHAR2(30)   := FND_API.G_MISS_CHAR
,   Attribute1                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute2                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute3                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute4                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute5                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute6                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute7                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute8                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute9                    VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute10                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute11                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute12                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute13                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute14                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute15                   VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Return_Status                 VARCHAR2(1)    := FND_API.G_MISS_CHAR
,   Transaction_Type              VARCHAR2(30)   := FND_API.G_MISS_CHAR
,   Original_System_Reference     VARCHAR2(50)   := FND_API.G_MISS_CHAR);
TYPE Assembly_Item_Tbl_Type IS TABLE OF Assembly_Item_Rec_Type
    INDEX BY BINARY_INTEGER;
```

Revised Components

```
TYPE Rev_Component_Rec_Type IS RECORD
(   Organization_CodeVARCHAR2(3):= FND_API.G_MISS_CHAR
,   Assembly_Item_NameVARCHAR2(81):= FND_API.G_MISS_CHAR
,   Start_Effective_DateDATE:= FND_API.G_MISS_DATE
,   Disable_DateDATE:= FND_API.G_MISS_DATE
,   Operation_Sequence_NumberNUMBER := FND_API.G_MISS_NUM
,   Component_Item_NameVARCHAR2(81):= FND_API.G_MISS_CHAR
,   Alternate_BOM_CodeVARCHAR2(10):= FND_API.G_MISS_CHAR
,   New_Effectivity_Date    DATE:= FND_API.G_MISS_DATE
,   New_Operation_Sequence_Number NUMBER := FND_API.G_MISS_NUM
,   Item_Sequence_NumberNUMBER:= FND_API.G_MISS_NUM
,   Quantity_Per_AssemblyNUMBER:= FND_API.G_MISS_NUM
,   Planning_PercentNUMBER:= FND_API.G_MISS_NUM
,   Projected_YieldNUMBER:= FND_API.G_MISS_NUM
,  Include_In_Cost_RollupNUMBER      := FND_API.G_MISS_NUM
,  Wip_Supply_TypeNUMBER      := FND_API.G_MISS_NUM
,  So_BasisNUMBER := FND_API.G_MISS_NUM
,  Optional                    NUMBER := FND_API.G_MISS_NUM
,  Mutually_ExclusiveNUMBER:= FND_API.G_MISS_NUM
,  Check_AtpNUMBER := FND_API.G_MISS_NUM
,  Shipping_Allowed            NUMBER        := FND_API.G_MISS_NUM
,  Required_To_Ship            NUMBER      := FND_API.G_MISS_NUM
,  Required_For_Revenue        NUMBER      := FND_API.G_MISS_NUM
,  Include_On_Ship_Docs        NUMBER      := FND_API.G_MISS_NUM
,  Quantity_Related            NUMBER      := FND_API.G_MISS_NUM
,  Supply_Subinventory         VARCHAR2(10):= FND_API.G_MISS_CHAR
,  Location_Name       VARCHAR2(81)   := FND_API.G_MISS_CHAR
,  Minimum_Allowed_Quantity   NUMBER:= FND_API.G_MISS_NUM
,  Maximum_Allowed_Quantity   NUMBER        := FND_API.G_MISS_NUM
,  component_remarks           VARCHAR2(240)  := FND_API.G_MISS_CHAR
,  Attribute_category          VARCHAR2(30)   := FND_API.G_MISS_CHAR
,  Attribute1                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute2                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute3                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute4                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute5                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute6                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute7                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute8                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute9                  VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute10                 VARCHAR2(150)  := FND_API.G_MISS_CHAR
,  Attribute11                 VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute12                 VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute13                 VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute14                 VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute15                 VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Return_Status                 VARCHAR2(1)    := FND_API.G_MISS_CHAR
,   Transaction_Type   VARCHAR2(30)  := FND_API.G_MISS_CHAR
,   Original_System_Reference    VARCHAR2(50)   := FND_API.G_MISS_CHAR
,   From_End_Item_Unit_Number     VARCHAR2(30)  := FND_API.G_MISS_CHAR
,   To_End_Item_Unit_Number        VARCHAR2(30)   := FND_API.G_MISS_CHAR);
TYPE Rev_Component_Tbl_Type IS TABLE OF Rev_Component_Rec_Type
    INDEX BY BINARY_INTEGER;
```

Reference Designator

```
TYPE Ref_Designator_Rec_Type IS RECORD
(   Organization_CodeVARCHAR2(3):= FND_API.G_MISS_CHAR
,   Revised_Item_NameVARCHAR2(81) := FND_API.G_MISS_CHAR
,   Start_Effective_DateDATE:= FND_API.G_MISS_DATE
,   Operation_Sequence_NumberNUMBER := FND_API.G_MISS_NUM
,   Component_Item_NameVARCHAR2(81):= FND_API.G_MISS_CHAR
,   Alternate_Bom_CodeVARCHAR2(10):= FND_API.G_MISS_CHAR
,   Reference_Designator_NameVARCHAR2(10) := FND_API.G_MISS_CHAR
,   Ref_Designator_CommentVARCHAR2(240):= FND_API.G_MISS_CHAR
,   Attribute_category        VARCHAR2(30)  := FND_API.G_MISS_CHAR
,   Attribute1               VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute2               VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute3               VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute4               VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute5               VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute6              VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute7              VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute8              VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute9              VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute10             VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute11             VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute12             VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute13             VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute14             VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   Attribute15             VARCHAR2(150)  := FND_API.G_MISS_CHAR
,   New_Reference_Designator  VARCHAR2(10)   := FND_API.G_MISS_CHAR
,   Return_Status           VARCHAR2(1)    := FND_API.G_MISS_CHAR
,   Transaction_TypeVARCHAR2(30)   := FND_API.G_MISS_CHAR
,   Original_System_Reference    VARCHAR2(50)   := FND_API.G_MISS_CHAR);
TYPE Ref_Designator_Tbl_Type IS TABLE OF Ref_Designator_Rec_Type
    INDEX BY BINARY_INTEGER;
```

Substitute Components

```
TYPE Sub_Component_Rec_Type IS RECORD
(   Organization_CodeVARCHAR2(3) := FND_API.G_MISS_CHAR
,   Revised_Item_NameVARCHAR2(81) := FND_API.G_MISS_CHAR
,   Start_Effective_DateDATE := FND_API.G_MISS_DATE
,   Operation_Sequence_Number NUMBER := FND_API.G_MISS_NUM
,   Component_Item_NameVARCHAR2(81) := FND_API.G_MISS_CHAR
,   Alternate_BOM_CodeVARCHAR2(10) := FND_API.G_MISS_CHAR
,   Substitute_Component_Name VARCHAR2(81) := FND_API.G_MISS_CHAR
,   Substitute_Item_Quantity  NUMBER          := FND_API.G_MISS_NUM
,   Attribute_category        VARCHAR2(30)    := FND_API.G_MISS_CHAR
, Attribute1            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute2            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute3            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute4            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute5            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute6            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute7            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute8            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute9            VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute10           VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute11           VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute12           VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute13           VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute14           VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Attribute15           VARCHAR2(150)   := FND_API.G_MISS_CHAR
, Return_Status         VARCHAR2(1)     := FND_API.G_MISS_CHAR
, Transaction_Type      VARCHAR2(30)    := FND_API.G_MISS_CHAR
, Original_System_Reference   VARCHAR2(50)   := FND_API.G_MISS_CHAR);
TYPE Sub_Component_Tbl_Type IS TABLE OF Sub_Component_Rec_Type
  INDEX BY BINARY_INTEGER;
```

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.

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_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.

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_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. 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 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
);
```

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 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
);
```

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 BH (Bill Header), 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 - BOM_BO_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_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.

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 - BOM_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_BOM

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 BOM_Header
4. If BOM Revisions records exist, call BOM_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: BOM_ Header, BOM_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.

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.

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: Bill of Material entry already exists.
2. 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.
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 ECO. Such errors affect the entire business object. All record statuses are set to ‘F’. The following are situations that cause this error:

7. You do not have access to this Item Type
8. 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.
9. 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.

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 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

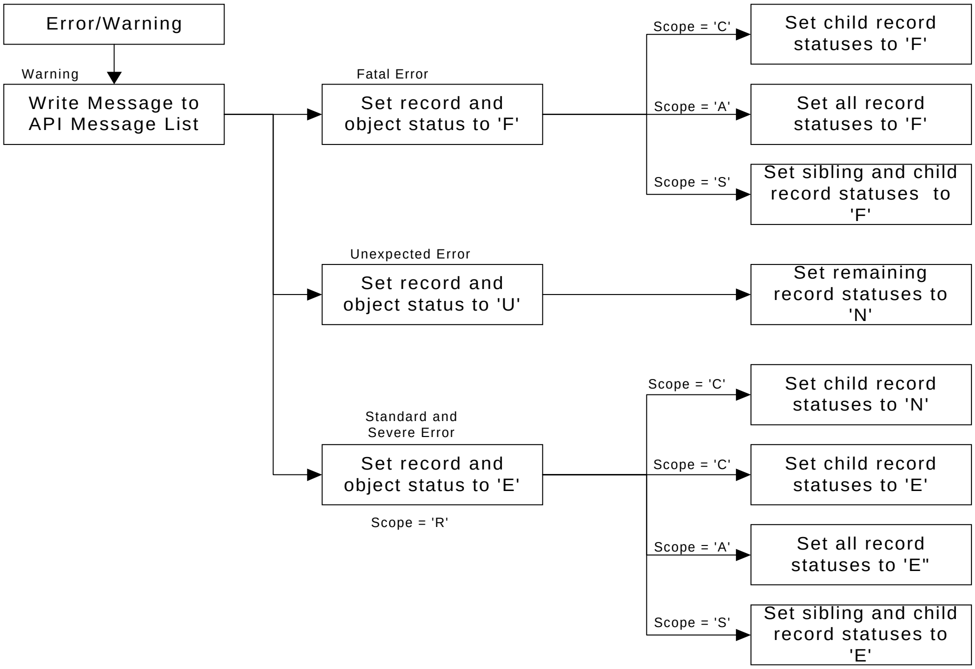
Record Error Status : This is the state of the record after the import (success or error). The error status is also referred to as *return status* or *error state* in this document. Please see the Error States section above for a list of statuses a record can receive. The error status helps locate erroneous 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:


Error/Warning
Warning
Write Message to API Message List
Fatal Error
Set record and object status to 'F'
Scope = 'C'
Set child record statuses to 'F'
Scope = 'A'
Set all record statuses to 'F'
Scope = 'S'
Set sibling and child record statuses to 'F'
Unexpected Error
Set record and object status to 'U'
Set remaining record statuses to 'N'
Standard and Severe Error
Set record and object status to 'E'
Scope = 'R'
Scope = 'C'
Set child record statuses to 'N'
Scope = 'C'
Set child record statuses to 'E'
Scope = 'A'
Set all record statuses to 'E"
Scope = 'S'
Set sibling and child record statuses to 'E'

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 status 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 status 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 status to ‘F’

p_severity_level = Unexpected Error

Log error messages
Set record status to ‘U’
Set all remaining un-processed business object record statuses to ‘N’
Set business object status to ‘U’
p_severity_level = Warning
Log warning messages

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.
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 - 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:

1. 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.
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 it's 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.
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 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:

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
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 organziation structures.

Launching the Export

You need to call the API in the following way:

1. `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 whi ch 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

4

Oracle Cost Management Open Interfaces

This chapter contains information about the following Oracle Cost Management open interfaces and application program interfaces:

- Periodic Cost Open Interface on page 4-2

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 User's Guide

Oracle Bill of Materials Technical 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:

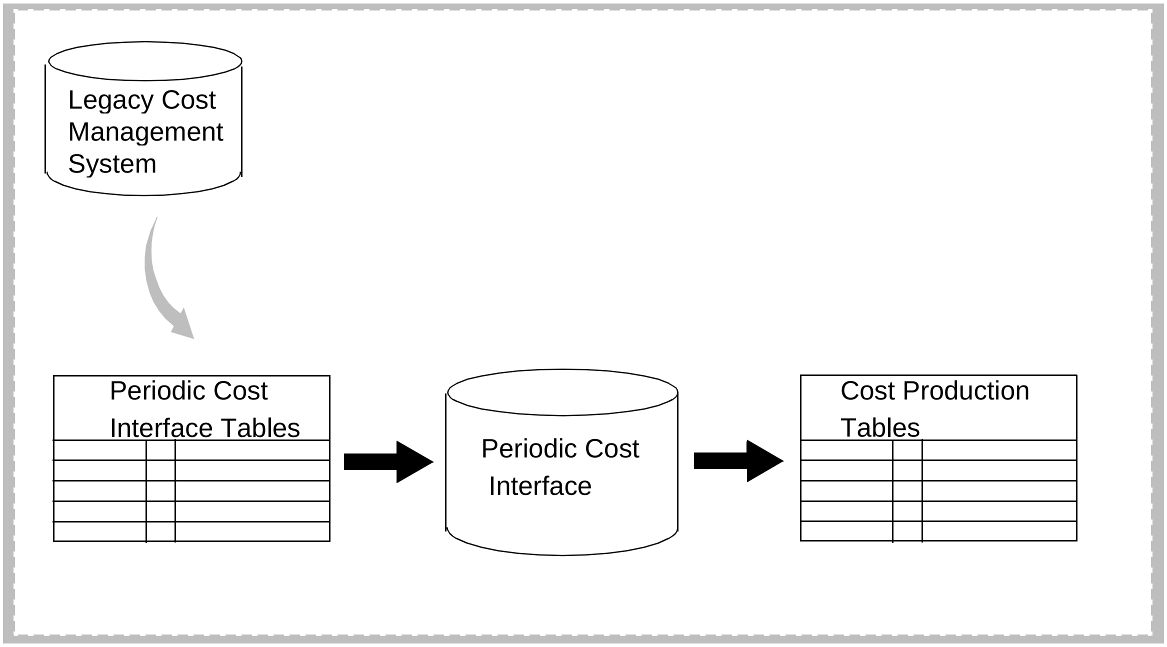
- CST_PAC_ITEM_COSTS
- CST_PAC_ITEM_COST_DETAILS
- CST_PAC_QUANTITY_LAYERS

See Also

Oracle Cost Management User's Guide

Oracle Bill of Materials Technical Reference Manual

Figure 4–1 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.

Table 4–1 Periodic Costing Open Costs Interface Table

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			

Table 4–1 Periodic Costing Open Costs Interface Table

CST_PC_ITEM_COSTS_INTERFACE					
Column Name	**Type**	**Required**	**Derived**	**Optional**	**Reserved for Future Use**
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				
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		
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		

Table 4–1 Periodic Costing Open Costs Interface Table

CST_PC_ITEM_COSTS_INTERFACE					
Column Name	**Type**	**Required**	**Derived**	**Optional**	**Reserved for Future Use**
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		
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				
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.

Table 4–2 Detail Table Periodic Cost Open Interface

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	
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 Technical Reference Manual.*

Required Data

The Periodic Cost Interface uses the PROCESS_FLAG to indicate whether processing of the row succeeded or errorred. 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 4–3

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
	COST_ELEMENT_ID	not null
	ITEM_COST	not null

Table 4–3

Table	Columns	Required Value
	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

See Table 4–1 and Table 4–2 for a list of desired columns during import. 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
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

Error Flag	Validation Rule	Error Explanation	Table
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
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

Error Flag	Validation Rule	Error Explanation	Table
13	All quantity and cost columns should have a value of >= 0	Cost Value Error	CPICI CPCDI
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

anirban can we say anything about the 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
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

Message Code	Message Text	Type
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

5

Engineering Change Order Business Object Interface

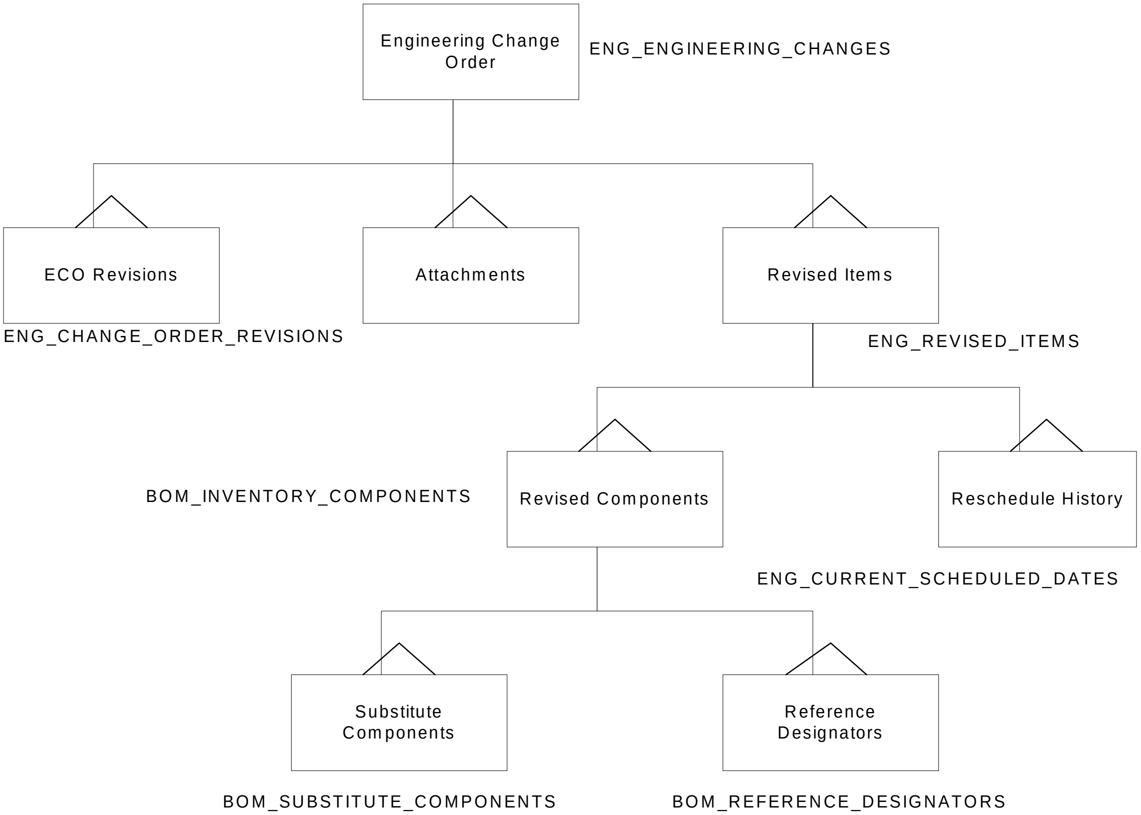
The Engineering Change Order (ECO) Business Object Interface allows you to import your change order information from a legacy or product data management (PDM) system into Oracle Engineering. You can create, update and delete ECO information. With the ECO Business Object Interface, you can automatically 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 on page 5-2
- Business Object APIs on page 5-6
- Entity Process Flows on page 5-13

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.



- 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.

Table 5–1

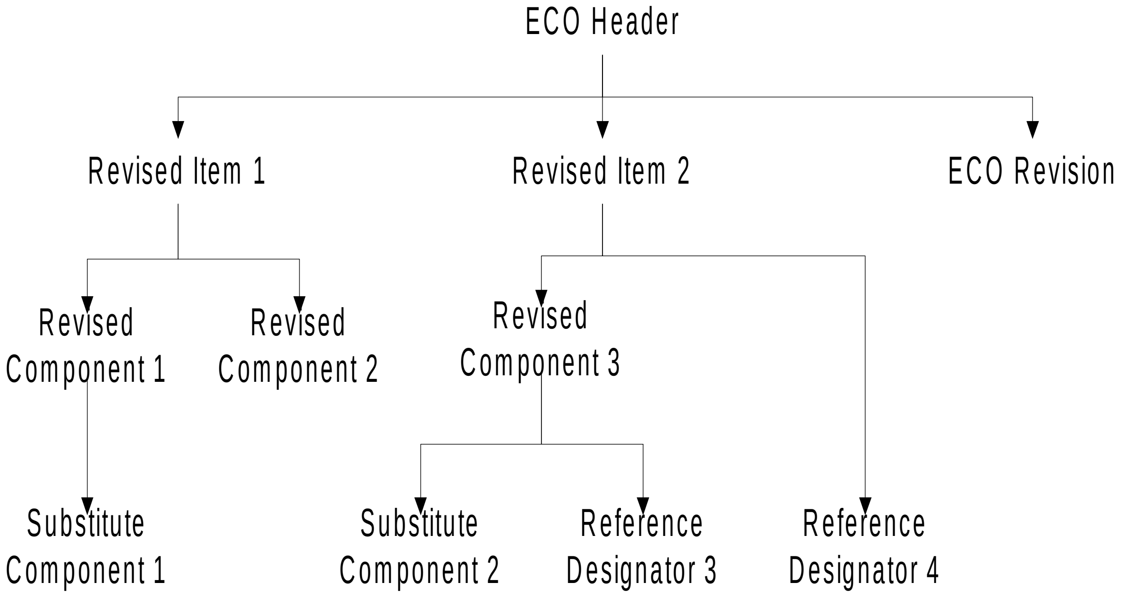
Production Table	Description
ENG_ENGINEERING_CHANGES	Stores information about ECO headers
ENG_CHANGE_ORDER_REVISIONS	Stores information about revisions of an ECO
ENG_REVISED_ITEMS	Stores information about revised items on an ECO
BOM_INVENTORY_COMPONENTS	Stores information about the un-implemented single-level BOM components

Table 5–1

Production Table	Description
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 it's 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 it's own. Here is an example:



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.

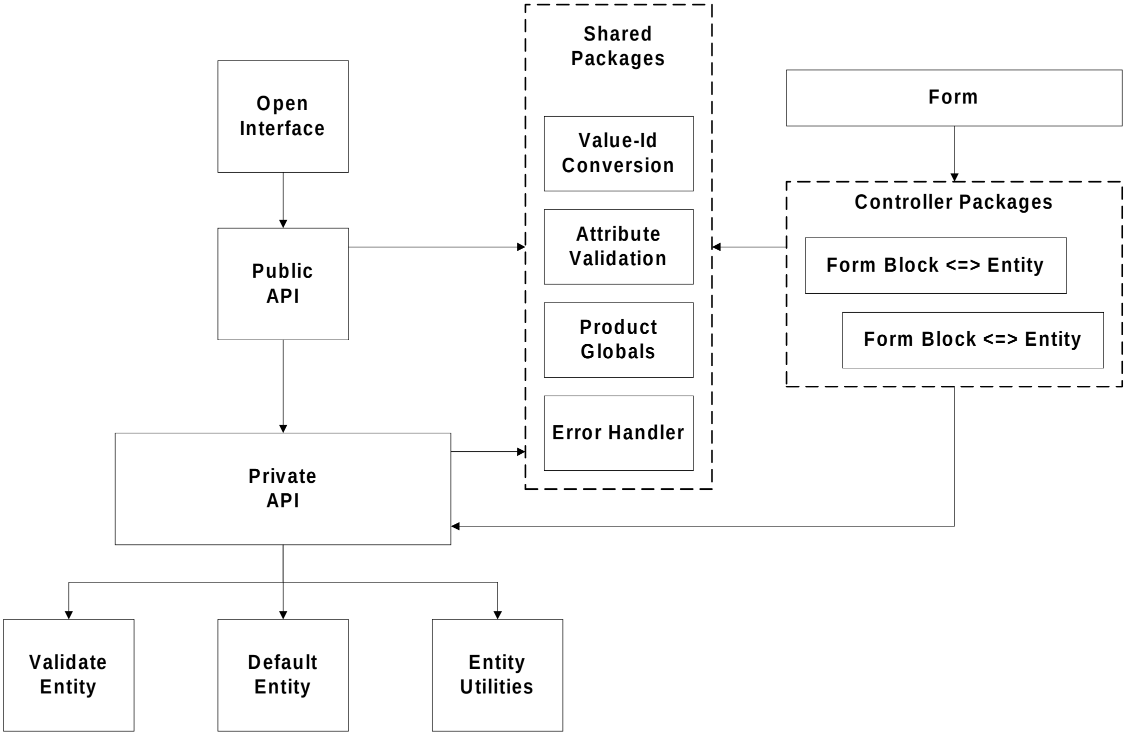
Substitute Component: The substitute components entity is a direct child of the revised components entity. It cannot exists without a revised component. You can enter information into this entity if you wish to add or disable a substitute component on your revised component. To identify a substitute 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.

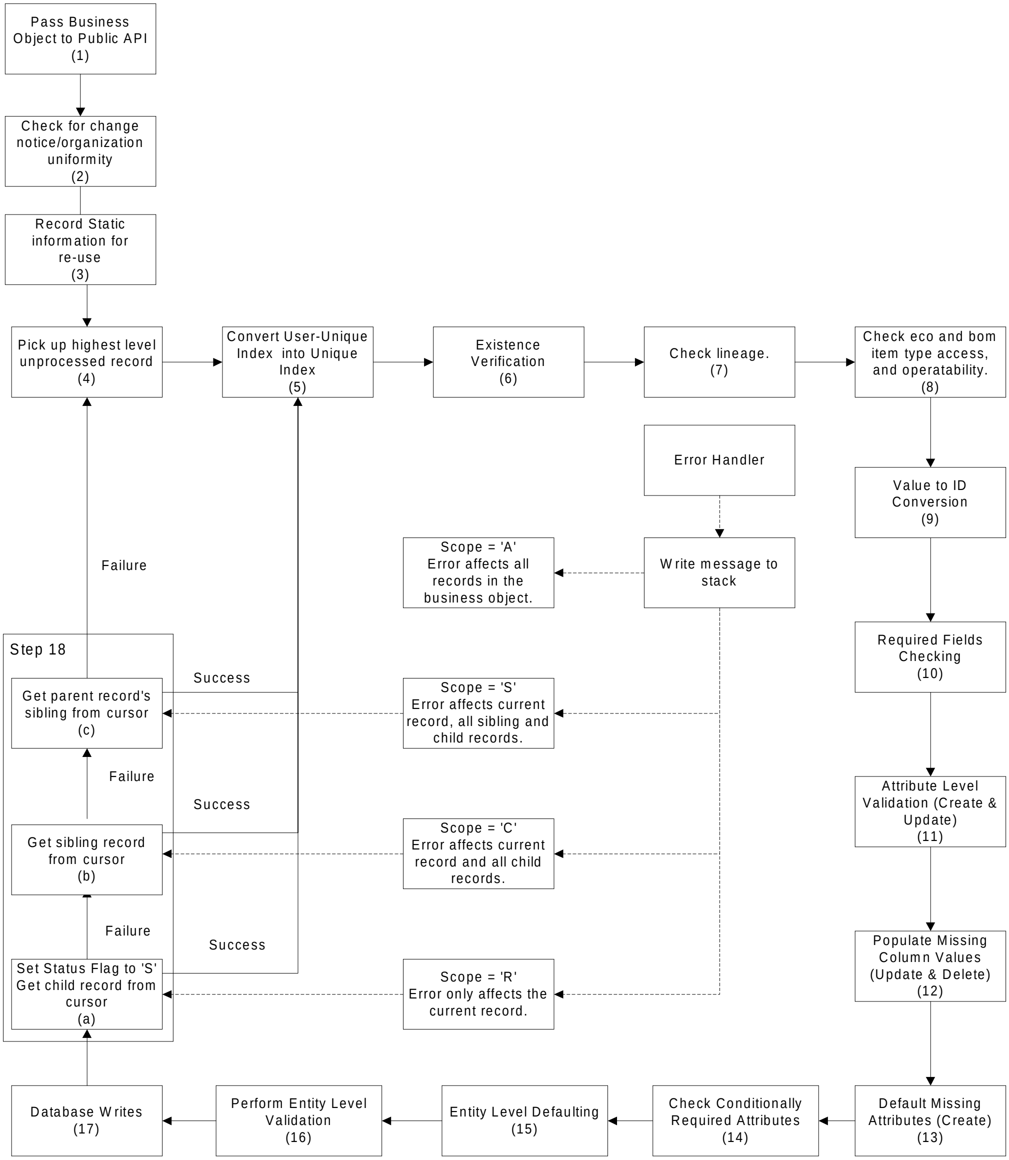
To import data into the ECO Business Object Interface, you simply need to categorize your data into ECO Business Object and identify which records you wish to import. No elaborate Ids or system specific information is required. The information you need to pass to the ECO Business Object Interface is 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.


Pass Business Object to Public API (1)
Check for change notice/organization uniformity (2)
Record Static information for re-use (3)
Pick up highest level unprocessed record (4)
Convert User-Unique Index into Unique Index (5)
Existence Verification (6)
Check lineage. (7)
Check eco and bom item type access, and operatability. (8)
Value to ID Conversion (9)
Required Fields Checking (10)
Attribute Level Validation (Create & Update) (11)
Populate Missing Column Values (Update & Delete) (12)
Default Missing Attributes (Create) (13)
Check Conditionally Required Attributes (14)
Entity Level Defaulting (15)
Perform Entity Level Validation (16)
Database Writes (17)
Step 18
Set Status Flag to 'S' Get child record from cursor (a)
Get sibling record from cursor (b)
Get parent record's sibling from cursor (c)
Failure
Failure
Failure
Success
Success
Success
Error Handler
Write message to stack
Scope = 'A' Error affects all records in the business object.
Scope = 'S' Error affects current record, all sibling and child records.
Scope = 'C' Error affects current record and all child records.
Scope = 'R' Error only affects the current record.

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 API Framework

The below framework demonstrates how forms and business object interfaces can use the business object API framework simultaneously.

Process Flow

Table 5–2

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 / 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 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

Table 5–2

Step	Purpose	Description	Error
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 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 it's 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 8(b): Check ECO operability	An ECO and any of it's 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 it's 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)

Table 5–2

Step	Purpose	Description	Error
Step 8(d): Check operability of parent items and current item (if applicable)	A revised item and any of it's 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 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 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

Table 5–2

Step	Purpose	Description	Error
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.	n For all NULL columns in record, try to default in values based on other values. n Set all MISSING column values to NULL.	CREATE: Severe Error IV. UPDATE: Standard Error
Step 16: Entity level validation	This is where the whole record is checked. The following are checked: n Non-updateable columns (UPDATEs): Certain columns must not be changed by the user when updating the record. n Cross-attribute checking: The validity of attributes may be checked, based on factors external to it. n 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: n Insert record for CREATE n Overwrite record for UPDATE and CANCEL n Remove record for DELETE	-N/A-

Table 5–2

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

Columns Exposed to the User

Table 5–3

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	

Table 5–3

Exposed Name	Actual Column Name	Type	Comments
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)	
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

Table 5–4

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)	
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Revised Items

Columns Exposed to the User :

Table 5–5

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	
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)	

Table 5–5

Name	Assoc. Column Name	Type	Comments
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)	

Notes:

- 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, it's 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

Table 5–6

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	
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)	

Table 5–6

Name	Associated Column Name	Type	Comments
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
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)	

Table 5–6

Name	Associated Column Name	Type	Comments
Attribute11	Attribute11	VARCHAR2(150)	
Attribute12	Attribute12	VARCHAR2(150)	
Attribute13	Attribute13	VARCHAR2(150)	
Attribute14	Attribute14	VARCHAR2(150)	
Attribute15	Attribute15	VARCHAR2(150)	

Notes :

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

Table 5–7

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
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)	

Table 5–7

Name	Associated Column Name	Type	Comments
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)	

Substitute Components

Columns Exposed to the User

Table 5–8

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

Table 5–8

Name	Name	Type	Comments
Transaction_Type	-N/A	VARCHAR2(30)	Required
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:

In order to use the business object APIs effectively, the user must follow the three step rule:

5. Initialize the system information.
6. Call the Public API.
7. 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

Table 5–9

Description	Cause of Failure	Error	Message
n 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. n 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.

Table 5–10

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.

Table 5–11

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. It's 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:

Table 5–12

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.

Table 5–13

CODE	MEANING
‘S’	Success
‘E’	Error

Table 5–13

CODE	MEANING
‘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.

Table 5–14

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:

Table 5–15

CODE	MEANING
‘E’	Error
‘N’	Not Processed

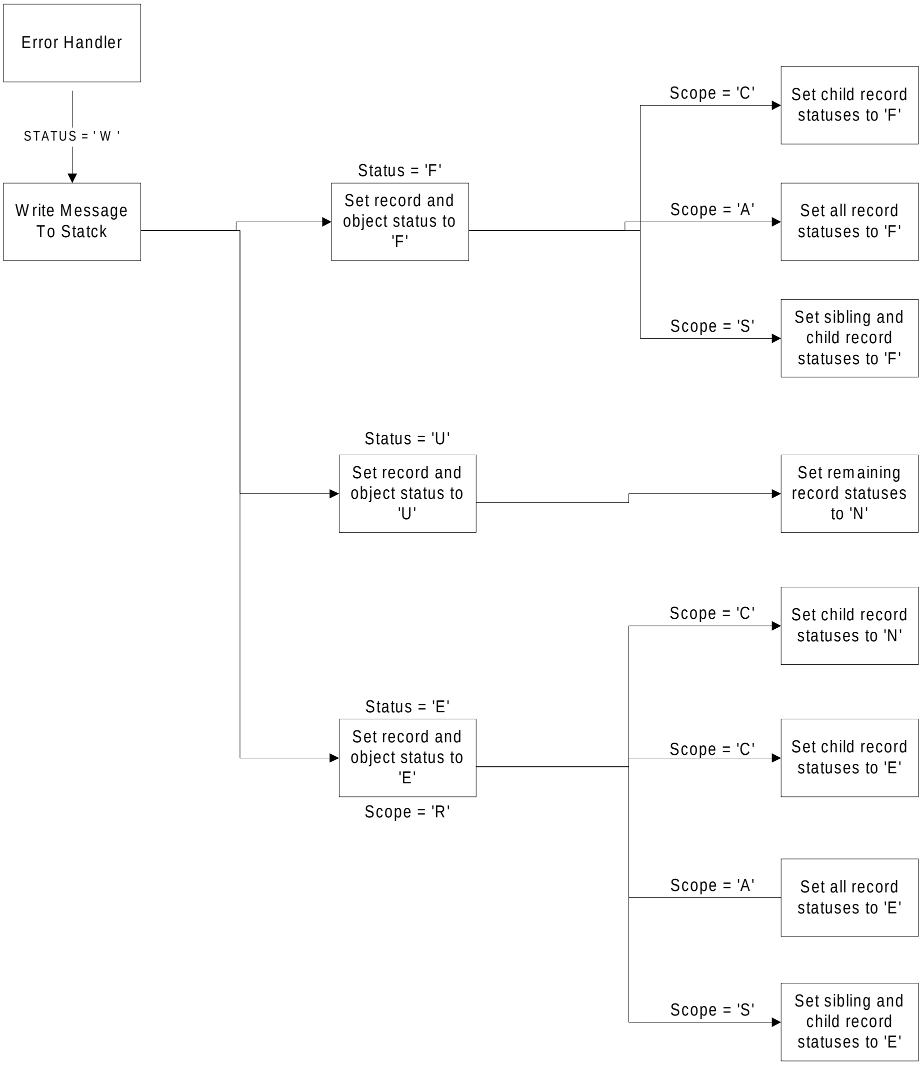
Error Classes

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.


Error Handler
STATUS = ' W '
Write Message To Statck
Status = 'F'
Set record and object status to 'F'
Scope = 'C'
Set child record statuses to 'F'
Scope = 'A'
Set all record statuses to 'F'
Scope = 'S'
Set sibling and child record statuses to 'F'
Status = 'U'
Set record and object status to 'U'
Set remaining record statuses to 'N'
Scope = 'C'
Set child record statuses to 'N'
Status = 'E'
Set record and object status to 'E'
Scope = 'R'
Scope = 'C'
Set child record statuses to 'E'
Scope = 'A'
Set all record statuses to 'E'
Scope = 'S'
Set sibling and child record statuses to 'E'

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 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 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 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.

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

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 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:

Table 5–16

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.

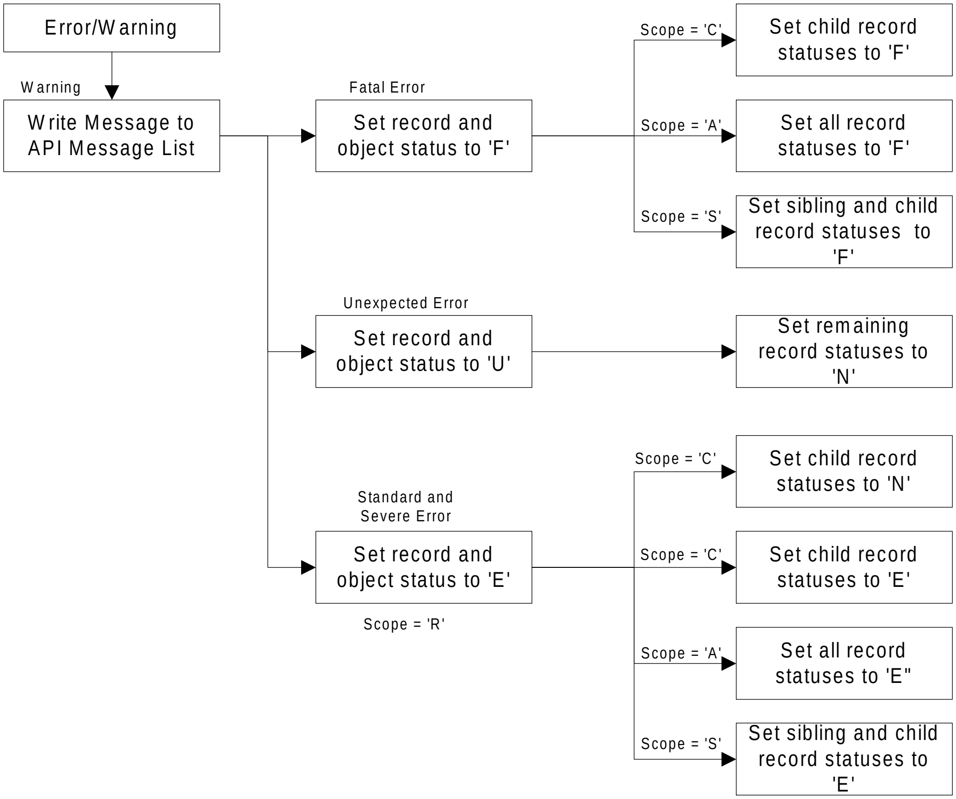
- Every record is assigned an error status by the import program. Hence, if a record has a NULL return status, it means that the import program has not gotten to it yet.
- The user must send in records with {return status = NULL}. The import program will not look at records that already have an error 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 :

Table 5–17

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:


Error/Warning
Warning
Write Message to API Message List
Fatal Error
Set record and object status to 'F'
Scope = 'C'
Set child record statuses to 'F'
Scope = 'A'
Set all record statuses to 'F'
Scope = 'S'
Set sibling and child record statuses to 'F'
Unexpected Error
Set record and object status to 'U'
Set remaining record statuses to 'N'
Scope = 'C'
Set child record statuses to 'N'
Standard and Severe Error
Set record and object status to 'E'
Scope = 'R'
Scope = 'C'
Set child record statuses to 'E'
Scope = 'A'
Set all record statuses to 'E"
Scope = 'S'
Set sibling and child record statuses to 'E'

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:

Table 5–18

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
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 it’s 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 occured in the revised item, the other message. This message essentially mentions the following:

3. 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.
4. which record caused this record to be affected.
5. what process flow step in the offending record caused this record to be affected.
6. 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 it’s 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:

Table 5–19

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.
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.

Table 5–20

Field	Description
Message Name	Name of the message used to retrieve the translated message text. NULL for unexpected errors.

Table 5–20

Field	Description
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.

6

Oracle Inventory Open Interfaces and APIs

This chapter contains information about the following Oracle Inventory open interfaces and application program interfaces:

Open Transaction Interface

Oracle Inventory provides an open interface for you to easily load transactions from external applications and feeder systems. These transactions could include sales order shipment transactions from an order entry system other than Oracle Order Entry, or they could be simple material issues, receipts, or transfers loaded from data collection devices. The following transaction types are supported by this interface:

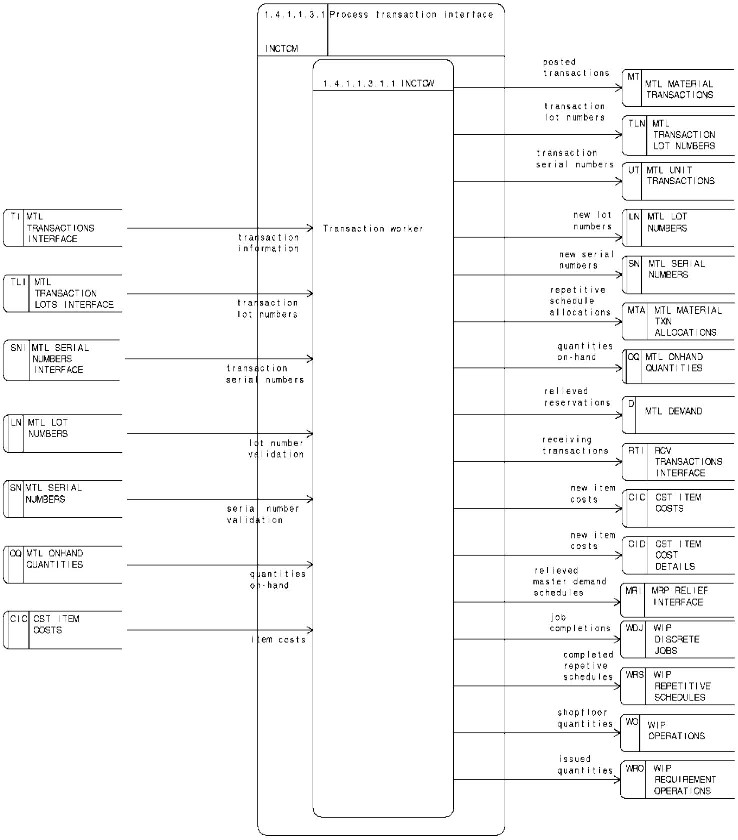
- Inventory issues and receipts (including user-defined transaction types)
- Subinventory transfers
- Direct inter-organization transfers
- Intransit shipments
- WIP component issues and returns
- WIP assembly completions and returns
- Sales order shipments
- Inventory average cost updates

This interface is also used as an integration point with Oracle Order Entry for shipment transactions. Oracle Order Management's Inventory Interface program populates the interface tables with transactions submitted through the Confirm Shipments window.

Functional Overview

The following data flow diagram shows the key tables and programs that comprise the Transaction Interface for Inventory Movement Transactions, WIP Issue and Completion Transactions, Sales Order Shipments, and Inventory Average Cost Update Transactions.

Figure 6–1 Open Transaction Interface



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.

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.

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 Entry 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 Entry 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

The following graphic describes the 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	
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		

Column Name	Type	Required	Derived	Optional
LOC_SEGMENT1toLOC_ SEGMENT20	Varchar2(40)	3		
TRANSACTION_QUANTITY	Number	x		
TRANSACTION_UOM	Varchar2(3)	x		
PRIMARY_QUANTITY	Number		x	
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		
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

Column Name	Type	Required	Derived	Optional
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		
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
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

Column Name	Type	Required	Derived	Optional
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
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
DEMAND_CLASS	Varchar2(1)			14
PARENT_ID	Number			14
SUBSTITUTION_ID	Number			14
SUBSTITUTION_ITEM_ID	Number			14
SCHEDULE_GROUP	Number			14
BUILD_SCHEDULE	Number			14

Column Name	Type	Required	Derived	Optional
REFERENCE_CODE	Number			14
FLOW_SCHEDULE	Varchar2(1)	16		
SCHEDULED_FLAG		17		

Notes:

[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 order-less completions
[15] For work order-less 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 Entry 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.

Note: See: Validation on page 6-26.

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:

Table 6–1 Transaction Types and Internal IDs

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

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.

Table 6–2 TRANSACTION_SOURCE_ID, Foreign Key References

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:

Table 6–3 Table Mapping: MTL_TRANSACTIONS_INTERFACE to MTL_DEMAND

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.

MTL_TRANSACTION_LOTS_INTERFACE

The following graphic describes the MTL_TRANSACTION_LOTS_INTERFACE table:

Table 6–4 Transaction Lot Numbers Interface

Column Name	Type	Required	Derived	Optional
TRANSACTION_INTERFACE_ ID	Number	3		
SOURCE_CODE	Varchar2(30)			x
SOURCE_LINE_ID	Number			x
LOT_NUMBER	Varchar2(30)	x		
LOT_EXPIRATION_DATE	Date	1		
TRANSACTION_QUANTITY	Number	x		
PRIMARY_QUANTITY	Number		x	
SERIAL_TRANSACTION_TEMP_ID	Number	2		
ERROR_CODE	Varchar2(30)		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

Notes:

[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.

MTL_SERIAL_NUMBERS_INTERFACE

The following graphic describes the MTL_SERIAL_NUMBERS_INTERFACE Interface table:

Table 6–5 Transaction Serial Numbers Interface

Column Name	Type	Required	Derived	Optional
TRANSACTION_INTERFACE_ ID	Number	x		
SOURCE_CODE	Varchar2(30)			x
FM_SERIAL_NUMBER	Varchar2(30)	x		
TO_SERIAL_NUMBER	Varchar2(30)			x
SOURCE_LINE_ID	Number			x
VENDOR_SERIAL_NUMBER	Varchar2(30)			x
ERROR_CODE	Varchar2(30)		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

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.

CST_COMP_SNAP_INTERFACE

The following graphic describes the CST_COMP_SNAP_INTERFACE Interface table:

Table 6–6 Completion Cost Calculation 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	
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.

Open Replenishment Interface

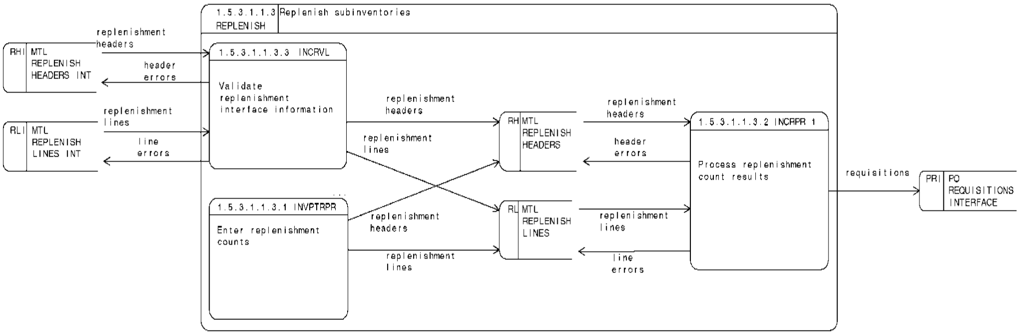
Oracle Inventory provides an open interface for you to easily load replenishment requests from external systems such as a bar-code application. Such requests may be in the form of stock-take counts or requisition requests for subinventories in which you do not track quantities.

You may also use the Replenishment Interface to process requisition requests generated by external applications for tracked subinventories.

Functional Overview

The following data flow diagram shows the key tables and programs that comprise the Replenishment Interface:

Figure 6–2



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*.

Setting Up the Replenishment Interface

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:

Table 6–7 Oracle Inventory Replenishment Headers Interface

Column Name	Type	Required	Derived	Optional
REPLENISHMENT_HEADER_ID	Number	3		
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			3
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

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 - Hold

2 - Pending

3 - Processing

4 - Error

5 - Completed

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_HEADER_ID

Enter a unique identifier for the replenishment count. This column is used to group the lines of a replenishment count with the header. You may use the sequence MTL_REPLENISH_HEADERS_S to obtain a unique identifier.

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. A null value here indicates that you do not want to consider planned supply when performing replenishment calculations.

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:

Table 6–8 Oracle Inventory Replenishment Lines Interface

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		
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.

Open Item Interface

You can import items from any source into Oracle Inventory and Oracle Engineering using the Item 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 Interface validates your data, insuring that your imported items contain the same item detail as items you enter manually in the Master Item window. See: Defining Items, *Oracle Inventory User's Guide*.

The purpose of this essay is to explain how to use the Item Interface.

Functional Overview

The Item Interface lets you import items into Oracle Inventory and, if installed at your site, Oracle Engineering. When you import items through the Item Interface, you create new items in your item master organization 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 Interface also lets you import revision details, including past and future revisions and effectivity dates. Validation of imported items is done using the same rules as the item definition windows, so you are insured of valid items. See: Overview of Engineering Prototype Environment, *Oracle Engineering User's Guide* and Defining Items, *Oracle Inventory User's Guide*.

The Item Interface reads data from two tables for importing items and item details. You use the MTL_SYSTEMS_ITEM_INTERFACE table for your new item numbers and all item attributes. This is the main item interface table, and may be the only table you choose to use. If you are importing revision details for your new items, you can use the MTL_ITEM_REVISIONS_INTERFACE table. This table is used only for revision information, and is not required. A third table, MTL_INTERFACE_ERRORS, is used for error tracking of all items that the Item Interface fails.

Before you use the Item Interface, you must write and run a custom program that extracts item information from your source system and inserts it into the MTL_SYSTEM_ITEM_INTERFACE table, and (if revision detail is included) the MTL_ITEMS_REVISIONS_INTERFACE table. After you load the items into these interface tables, you run the Item Interface to import the data. The Item Interface assigns defaults, validates data you include, and then imports the new items.

> **Note:** You must import items into a master organization before you import items into additional organizations. You can accomplish this by specifying only your master organization on a first pass run of the Item Interface. Once this has completed, you can run the Item Interface again, this time specifying an additional or all organizations.

You can also use the Item Interface to import item material cost, material overhead, and revision details.

Setting Up the Item Interface

Create Indexes for Performance

You should create the following indexes to improve Item Interface performance.

First, determine which segments are enabled for the System Items flexfield.

Then, for example, if you have a two-segment flexfield, with segment8 and segment12 enabled, you would do the following:

```
SQL> create unique index MTL_SYSTEM_ITEMS_UC1 on mtl_system_items (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.

Start the Concurrent Manager

Since you launch and manage the Item 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: Oracle Inventory Profile Options, *Oracle Inventory User’s Guide* and Overview of Inventory Setup, *Oracle Inventory User’s Guide*.

Item Interface Runtime Options

To run the Item Interface, select Import Items from 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 Interface, you are prompted for report parameters. These are runtime options for the Item Interface:

All Organizations

Yes Run the interface for all organization codes in the item interface table.

No Run the interface only for the organization you are currently in. Item interface rows for organizations other than your current organization are ignored.

Validate Items

Yes Validate all items and their data residing in the interface table that have not yet been validated. If items are not validated, they will not be processed into Oracle Inventory.

No Do not validate items in the interface table. Note that items that have not been validated will not be processed into Oracle Inventory. You would use this option if you had previously run the item 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 interface tables.
No	Leave all rows in the item interface tables.

Process Set

Enter a number for the set id for the set of rows you want to process. The program picks up the rows marked with that id in the SET_PROCESS_ID column. If you leave this field blank, all rows are picked up for processing regardless of the SET_PROCESS_ID column value.

Inserting into the Item Interface Table

Item Interface Table Description

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 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 may also put in details about other interface tables not used by the Item Interface.

As currently running, the interface does not support the MTL_CROSS_REFERENCE_INTERFACE, MTL_ITEM_CATEGORIES_INTERFACE, or MTL_SECONDARY_LOCS_INTERFACE.

The MTL_ITEM_CATEGORIES_INTERFACE is used by the Item interface internally, but should not be populated by the user.

Table 6–9 Partial List of Columns, Oracle Inventory Item Interface

MTL_SYSTEM_ITEMS_INTERFACE				
(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		
SET_PROCESS_ID	Number	x		

Note: For information about columns not discussed in the Interface Manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual*.

Required Data

Every row in the item interface table must identify the item and organization. To identify the item when importing it, you may specify either the ITEM_NUMBER or SEGMENT*n* columns—the Item 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 has been entered and they conflict, ITEM_NUMBER overrides SEGMENT*n* and ORGANIZATION_ID overrides ORGANIZATION_CODE. It is strongly recommended 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, 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 dash (-) as your separator, a typical item would be entered as 1234-5678. When the Item Interface derives the item's segment values, 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, 5678 in SEGMENT2.

Note: If you enter values for SEGMENT*n* columns, be sure 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 Interface. Also, the Item 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 are also required to specify the DESCRIPTION. This has to 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. Of course, if the description is at the item-organization level, you are always able to override the master organization description by giving this column a value.

There are two other columns the Item Interface uses to manage processing. They are TRANSACTION_TYPE, which tells the Item Interface how to handle the row, and PROCESS_FLAG, which indicates the current status of the row.

Always set the TRANSACTION_TYPE 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 Interface.

The Item 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 Interface can pick up the row and process it into the production tables.

Table 6–10 Meaning of PROCESS_FLAG Values

Code	Meaning
1	Pending
2	Assign complete
3	Assign/validation failed
4	Validation succeeded; import failed
5	Import in process
7	Import succeeded

A full list of values for the PROCESS_FLAG is in Table 1–14, but you are unlikely to see all of these.

Other columns, although required in the production tables, are not required in the item interface table, because they have default values or their values can be derived from other sources. Check the defaults and derived values carefully, as they may not be the values you desire.

If the Item Interface successfully processes a row in the item interface table or the revision interface table, the program sets the PROCESS_FLAG to **7** (Import succeeded) for the row. If the Item Interface cannot insert a row into the production table, the PROCESS_FLAG column for the failed row is set to **4** (Import failed). If a row in the interface table fails validation, 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 Interface uses when you leave that column null in the item interface table. Columns with defaults are listed in Table 6–11 .

Table 6–11 Column Defaults in the Item Interface

MTL_SYSTEM_ITEMS_INTERFACE		
(Partial List of Columns) Column Name	**Default Value**	**Value Displayed in Window**
SUMMARY_FLAG [1]	Y	
ENABLED_FLAG	Y	
PURCHASING_ITEM_FLAG	N	No

Table 6–11 Column Defaults in the Item Interface

MTL_SYSTEM_ITEMS_INTERFACE		
(Partial List of Columns) Column Name	**Default Value**	**Value Displayed in Window**
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

Table 6–11 Column Defaults in the Item Interface

MTL_SYSTEM_ITEMS_INTERFACE		
(Partial List of Columns) Column Name	**Default Value**	**Value Displayed in Window**
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	*from **profile*** INV: Default Item Status	*from* Personal Profile Values
INVENTORY_PLANNING_CODE	6	Not planned
PLANNING_MAKE_BUY_CODE	2	Buy

Table 6–11 Column Defaults in the Item Interface

MTL_SYSTEM_ITEMS_INTERFACE		
(Partial List of Columns) Column Name	**Default Value**	**Value Displayed in Window**
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

Table 6–11 Column Defaults in the Item Interface

MTL_SYSTEM_ITEMS_INTERFACE		
(Partial List of Columns) Column Name	**Default Value**	**Value Displayed in Window**
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	

Notes:

[1] Defaulted to Y by the Item Interface, but the Master Items window defaults N for this column.

[2] Defaulted to N by the Item 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 when you have implemented a descriptive flexfield for items. To do this, simply include values for the descriptive flexfield columns (ATTRIBUTE_CATEGORY and ATTRIBUTE*n* columns) in the item interface table. No validation is performed on descriptive flexfield values.

In addition, the Item Interface uses the item’s status (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 item’s status, and any value in the attribute column of the item interface table is ignored. If an attribute is under default status control, then the attribute value derives from the item’s status only if there is no value in the attribute column of the item interface table. If an attribute is not under any status control, then the item status has no effect on the attribute’s value for the imported item.

Note: If an attribute is under status control, 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, 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), the Item 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 interface table. If you supply a value for a item-level attribute in a child organization record, the Item Interface rejects the record as an error. The exception is status attributes under status control. These attributes *always* derive from the item's status, never from the master record. See Table 6–11 for the list of defaults supplied by the Item Interface.

Whether you import a new item to an master organization or assign an existing item to a non-master organization, the Item Interface always enters a unique numeric identifier in the TRANSACTION_ID column of the item interface table, and the concurrent request number in the REQUEST_ID column of the item master table.

Item Categories

When the Item 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 Interface does not allow you to assign items to other category sets, nor does the interface allow you to specify an item's category value. See: Defining Category Sets, *Oracle Inventory User's Guide* and Defining Default Category Sets, *Oracle Inventory User's Guide*.

For example, suppose 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 using the Item Interface to assign an existing item to another organization, the item is also assigned to mandatory category sets with the default category. As described above, the item defining attributes determine to which mandatory category sets the item is assigned. Even if the item is assigned to a item level category set (non-mandatory) in the master organization, it is not assigned to that category set in the item’s new organization.

Validation

When you import an item, the Item Interface validates the data and any derived values the same way 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. You cannot import an item into both a master organization and a child organization at the same time. If you populate the item interface table with both master and child item records, you should run the Item Interface for the master organization only. After it has successfully finished running, run the interface again for all organizations. This second run inserts the items into the child organizations. See: Defining Items, *Oracle Inventory User’s Guide*

When you import items, the Item 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 Interface then validates each row. If a row in the interface table fails validation, 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 Oracle Inventory’s item master table, MTL_SYSTEM_ITEMS. If a row cannot be inserted into the item master table, 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 runtime option, the PROCESS_FLAG is set to 7 (Import Succeeded).

To minimize the number of rows 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, 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 Interface. You can also run multiple Item Interface processes. See: Multi-Thread Capability below.

Importing Additional Item Details

You can import additional cost and revision details for an imported item using interface tables listed in Table 6–12. The Item Interface imports the details specified in these tables at the same time that it imports the items themselves. The program validates all rows you insert into the interface tables, derives additional columns, and creates the item and item details in Oracle Inventory.

Table 6–12 Oracle Inventory Item Details Interface

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 there are many other tables in Oracle Inventory whose names may imply use, the tables listed in Table 6–12 are the *only* interface tables used by the Item Interface to import item details.

Before importing additional item details, you must complete the same setup steps required for manually defining these item details. For example, you must define your cost types and activities before you can assign item costs. The default cost category set must be specified using the Default Category Sets window, and the starting Revision must be set for all organizations. See: Overview of Inventory Setup, *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 Interface validates all required data and some optional data 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 Oracle Inventory validates manually entered details.

Importing Cost Details

When the Item Interface imports an item, it may also import costing information into Oracle Cost Management tables. The interface may import this costing information automatically using organization and category defaults, or you may specify the information for the item itself in the item interface table.

If you set up a default material overhead rate for the item's organization or for the default cost category, 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 Sub-Elements, *Oracle Cost Management User's Guide* and Defining Overhead, *Oracle Cost Management User's Guide*.

You may specify one material cost and one material overhead rate for the item directly in the item interface table itself. Remember to include the material sub-element for the material cost and overhead rate by specifying the sub-element.

The interface imports the basis type for the material sub-elements and material overhead subelements from the BOM_RESOURCES table. If the default basis type is not defined, the basis type of these sub-elements is set to *Item* and *Total Value* respectively.

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 interface table
- Do not specify any revisions and let the Item Interface default the revision based on the starting revision 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 may also include ECN information (see Table 6–13). You need to create your own program for populating this table.

Since 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 less than or equal to revision B, and so on. Each row in this table must correspond to a row in the item interface table. Each row must reference the item’s ITEM_NUMBER and ORGANIZATION_ID or ORGANIZATION_CODE.

Note: When importing multiple revisions for the same item, if one of the revisions fails validation, 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 interface table. The Item Interface automatically creates this revision with an effective date equal to the system date when it imports the item. (Use the revision interface table described above 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 interface table, the Item Interface assigns each item a beginning revision, using the default specified in the Organization Parameters. The system date is the effectivity date. Once established, you cannot add revisions with effectivity dates earlier than the date assigned by the Item Interface.

Note: Although most item information defaults from the master organization when you assign an existing item to a child organization, the Item Interface does *not* default an item’s revision detail from the master organization. See: Defining Item Revisions, *Oracle Inventory User’s Guide*.

As with the item 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 Table 6–10.

When you insert rows into the revision interface table, you should set the PROCESS_FLAG to 1 (Pending) and TRANSACTION_TYPE to CREATE.

Table 6–13 Column-Mappings from Revision Interface Table to Oracle Inventory

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, simply include values for the descriptive flexfield columns (ATTRIBUTE_CATEGORY and ATTRIBUTE*n* columns) in the revision interface table when you import revisions.

The Item Interface may also be used to create revisions for existing items. Only revision labels and effectivity dates higher than existing revisions may be imported. To do this, simply load revision detail for the existing items into MTL_ITEM_REVISIONS_INTERFACE and run the Item Interface.

Resolving Failed Interface Rows

If a row fails validation, the Item 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 interface table. For example, if a row in the item interface table fails, the program inserts a row into the interface errors table with the failed row’s TRANSACTION_ID as the error row’s TRANSACTION_ID. Each error in the interface errors table has a value for the MESSAGE_NAME and REQUEST_ID columns. The Item 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.

You should resolve errors in the sequence that 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 Interface will cause the other (spurious) errors for that failed row to disappear.

The Item Interface inserts validated rows into the production tables in Oracle Inventory and Oracle Cost Management. Depending on the item information you import, the interface inserts these rows into the item master, item categories, item costs, cost details, item revisions, pending item status, and uom conversions tables. If a row cannot be inserted into one of these tables, the PROCESS_FLAG column for all remaining rows is set to 4 (Import failed) and the Concurrent Item Interface inserts a row in the interface errors table. The program handles these processing errors in the same way 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 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 Inventory.

Resubmitting an Errored Row

During Item 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.

Table 6–14 Oracle Inventory Item Details Interface

PROCESS _FLAG	Error After and Before PROCESS_FLAG	Relaunch the Item 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
Notes: [1]MTL_SYSTEM_ITEMS_INTERFACE [2]MTL_INTERFACE_ERRORS [3]MTL_ITEM_CATEGORIES_INTERFACE [4]MTL_ITEM_REVISIONS_INTERFACE		

When you encounter rows errored out due to validations, you must first fix the row corresponding to the error with the appropriate value. Then reset PROCESS_FLAG = 1, INVENTORY_ITEM_ID = null, and TRANSACTION_ID = null. Then resubmit the row for reprocessing.

Multi-Thread Capability (Parallel Runs of the Item 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 above.

To have parallel runs of the Item Interface, the SET_PROCESS_ID column for records in the interface tables has to be populated with a positive, nonzero number.

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 Interface processes to accomplish this task.

In the scripts 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.

> **Note:** If you have custom scripts to insert data into the MTL_SYSTEM_ITEMS_INTERFACE table, 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 should have the matching SET_PROCESS_ID values.

From the Item Interface SRS launch form, specify the Process Set for a given run in the Process Set parameter. This initiates four Item Interface concurrent programs with the Process Set parameter set to 1, 2, 3, and 4 respectively. The four Item Interface processes run in parallel, each working on the set 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 multi-thread capability, 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 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.

Multi-threading Rules

The Applications DBA for the site should enforce these rules:

- If you run an Item Interface process with the Process Set value null, you should not concurrently run any other Item Interface processes.
- Do not have parallel runs of the Item Interface on the same process set.

Not following these rules will cause multiple Item Interface processes trying to work on the same set of rows and lead to unpredictable errors.

Customer Item and Customer Item Cross-Reference Open Interfaces

A number of manufacturing industries are characterized by 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 be able to 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 Interface and the Customer Item Cross-Reference Interface. These interfaces validate all data that you import into Oracle Inventory. They also perform foreign key validation and check for attribute inter-dependencies, acceptable values, and value ranges. The interfaces ensure that the imported customer items and cross-references contain the same detail as items 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 Interface

The Customer Item 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. Customer Address is required if you set Item Definition Level 3 while Customer Category is required for Item Definition Level 2. 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, you run the Customer Item Interface. The Customer Item Interface reads each record from the interface table and adds items that are successfully validated to the MTL_CUSTOMER_ITEMS table. Validation of the customer items uses the same rules as 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. You 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. Once a customer item cross-reference to an Inventory item has been defined, it is applicable to all organizations 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. Validation of the customer items cross-references uses the same rules as the Customer Item Cross-References window to ensure that only valid cross-references are imported.

Attention: The Customer Item Interface must be run successfully before the Customer Item Cross-Reference Interface. This is to ensure that a customer item has been 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 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 you include, and then import the new customer items and customer item cross-references.

Interface Runtime Options

You can access the Customer Item and Customer Item Cross-Reference Interfaces via 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 runtime: Abort on Error and Delete Successful Records.

Abort on Error

Valid values for this option are Yes or No. The default is No.

Yes - Both the Customer Item Interface and the Customer Item Cross-Reference Interface will abort execution if an error is encountered during validation of a record. No additional records will be 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 encountered during validation of records in the Customer Item 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 or No. The default is Yes.

Yes - Successfully validated records in the Customer Item 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 Interface and Customer Item Cross-Reference Interface simply populate the MTL_CUSTOMER_ITEMS and MTL_CUSTOMER_ITEM_XREFS tables without deleting records from the interface tables.

Customer Item Interface Table

Table Description

The Customer Item 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 need to be populated for audit trail and maintenance purposes can be found in the Table Administration and Audit Trail section. See: on page 6-72.

For information about columns not discussed in the Interface manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual*.

Table 6–15 List of Columns, Customer Item Interface

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	
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

Table 6–15 List of Columns, Customer Item Interface

Field Name	Type	Required
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
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	
MIN_FILL_PERCENTAGE	Number	
DEP_PLAN_REQUIRED_FLAG	Varchar2(1)	
DEP_PLAN_PRIOR_BLD_FLAG	Varchar2(1)	
INACTIVE_FLAG	Varchar2(1)	x

Table 6–15 List of Columns, Customer Item Interface

Field Name	Type	Required
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 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, it is strongly recommended that the CUSTOMER_ITEM_DESC field be populated with accurate information to clearly identify customer items by their description.

Customer

You define a customer by populating either of the CUSTOMER_NAME, CUSTOMER_NUMBER, or CUSTOMER_ID fields. Note that at least one of these fields must be entered for validation. The information provided in these fields is validated against the Oracle table RA_CUSTOMERS. The Interface will error 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 allows you to select multiple customer addresses by specifying the Address Category. The Address Category can be defined in the Interface tables by populating either one of the CUSTOMER_CATEGORY_CODE or the CUSTOMER_CATEGORY fields. 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 entered in these fields is validated against the RA_ADDRESSES table.

If both fields are populated to define an Address Category, 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 Oracle’s RA_ADDRESSES table, including any blank spaces, special characters and capitalized alphabets. The customer address you enter must exactly match the information in the RA_ADDRESSES table 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 different levels: Customer level, Address Category level or Address level.

A customer item 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, 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, or 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 the 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 Interface - Other fields

Transaction_Type

TRANSACTION_TYPE is a required field. The interface will error 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 supported currently by this interface.

Model items

You can define a customer item as a model item that can be referenced by other customer items. In order 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*.) Once a model customer item has been 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 will error 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 fashion 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 up 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 belonging 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, 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 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 Interface. Similarly, you must specify the master container for each detail container. The interface will error 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 fields. Alternatively, you can use the MASTER_CONTAINER_SEGMENTn field to specify a multi-segment 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 either the DETAIL_CONTAINER, DETAIL_CONTAINER_ITEM_ID or the SEGMENTn values for the descriptive flexfield. The same rules of precedence apply as for master container above.

Warning: The interface derives each item's segment values 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 DETAIL_CONTAINER fields for a multi-segment key flexfield.

Warning: If you enter values for SEGMENTn columns, 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 since no history information exists in the demand tables.

If a customer order falls outside the range defined by these fields then the demand processor will raise an exception to that order. This feature is designed to flag any order entry or EDI transfer errors, and to draw your attention towards 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 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 Interface table. You can also refer to the previous section for explanation of Descriptive Flexfields, PROCESS_FLAG and PROCESS_MODE, ERROR_CODE and ERROR_EXPLANATION, INACTIVE FLAG, and TRANSACTION_TYPE columns. See: Customer Item Interface Table on page 6-60.

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: on page 6-72.

Note: For information about columns not discussed in the Interface manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual*.

This section provides an explanation of fields used to define the Inventory Item, Master Organization, and Rank in the MTL_CI_XREFS_INTERFACE table.

Table 6–16 List of Columns, Customer Item Cross-Reference Interface

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			
TRANSACTION_TYPE	Varchar2(6)	x		
CUSTOMER_NAME	Varchar2(50)	Conditionally		
CUSTOMER_NUMBER	Varchar2(30)	Conditionally		

Table 6–16 List of Columns, Customer Item Cross-Reference Interface

Field Name	Type	Required	Derived	Optional
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		
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)			

Table 6–16 List of Columns, Customer Item Cross-Reference Interface

Field Name	Type	Required	Derived	Optional
ATTRIBUTEn	Varchar2(150)			
ERROR_CODE	Varchar2(9)			
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 multi-segment 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. This 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 will be processed before another item with a rank of 2 that references the same Customer Item. This is a required field and must be entered.

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 access the Cycle Count Entries Interface via the Reports, All menu in Oracle Inventory. The Interface offers the following options at runtime:

- 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 on page 6-77.

Note: For information about columns not discussed in the Interface manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual.*

Table 6–17 List of Columns, Cycle Count Entries Interface

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
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

Table 6–17 List of Columns, Cycle Count Entries Interface

Field Name	Type	Null
ACCOUNT_SEGMENT1-30	Varchar2(25)	Y
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
PROJECT_ID	Number	Y
TASK_ID	Number	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: on page 6-72.

Note: For information about columns not discussed in the Interface manual, see Table and View Definitions, *Oracle Inventory Technical Reference Manual*.

Table 6–18 List of Columns, Cycle Count Interface Errors

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.

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

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

See Also

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

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

Parameter Descriptions

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			
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

See Also

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

Lot Application Program Interface

The Lot API is a public API that allows you to insert a lot into the MTL_LOT_NUMBERS table. It performs all the necessary validation before it inserts the lot. This API derives the expiration date from the controls set for shelf_life_code and shelf_life_days for the item. If the API is able to insert the lot, it returns a status of success. If the lot already exists for the same item and organization, the API also returns a status of success; however, it places a message on the stack to indicate that the lot already exists. The Lot API returns an error status if there is any validation error. Standard WHO information is used from the FND_GLOBAL API. The Lot API has one public procedure, insertlot.

Setting Up the Lot API

Parameter Descriptions

The following chart lists all parameters used by the insertlot procedure. Additional information on these parameters follows the chart.

INSERTLOT

Parameter	Usage	Type	Required	Derived	Optional
p_inventory_item_id	IN	Number	x		
p_organization_id	IN	Number	x		
p_lot_number	IN	Varchar2	x		
p_expiration_date	IN/OUT	Date		x	x
x_return_status	OUT	Varchar2		x	

p_inventory_item_id

Indicates the inventory item identifier.

p_organization_id

Indicates the organization identifier.

p_lot_number

Indicates the lot number.

p_expiration_date

Indicates the expiration date.

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 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

See Also

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

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

Functional Overview

The Material Reservation API provides the following public procedures that allow you to accomplish the tasks listed above:

- query_reservation

  Returns all reservations that match the specified criteria.

- 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.

- delete_reservation

  Deletes existing reservations. Used when a reservation is no longer needed or has been fulfilled.

Setting Up the Material Reservation API

Parameter Descriptions

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		
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. 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_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			
p_rsv_rec	IN	Record	x		
p_serial_number	IN	PL/SQL Table			x
x_serial_number	OUT	PL/SQL Table			
p_partial_reservation_flag	IN	Varchar2			
p_force_reservation_flag	IN	Varchar2			
x_quantity_reserved	OUT	Number			
x_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. 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_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.

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			
p_to_serial_number	IN	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 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_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

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	Varchar2			
x_msg_data	OUT	Varchar2			
p_is_transfer_supply	IN	Varchar2	x		
p_original_rsv_rec	IN	Record	x		
p_original_serial_number	IN	Number			x
p_to_serial_number	IN	Number			x
p_validation_flag	IN	Varchar2			

Parameter	Usage	Type	Required	Derived	Optional
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. 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

x_to_reservation_id

Indicates the new reservation identifier.

DELETE_RESERVATION

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	
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. 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_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

See Also

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

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

Parameter Descriptions

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	
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

Condition	Message Returned	Meaning of Message Returned
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*.

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.

Functional Overview

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

Parameter Descriptions

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		

Parameter	Usage	Type	Required	Derived	Optional
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			
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

Indicates the order number, which is not unique in the Oracle Order Management system.

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			

Parameter	Usage	Type	Required	Derived	Optional
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

See Also

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

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)

Functional Overview

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

Parameter Descriptions

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	

Parameter	Usage	Type	Required	Derived	Optional
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_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.

CREATE_MOVE_ORDER_LINES

Parameter	Usage	Type	Required	Derived	Optional
p_api_version_number	IN	Number	x		
p_init_msg_list	IN	Varchar2		x	

Parameter	Usage	Type	Required	Derived	Optional
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_val_tbl	IN	PL/SQL Table	x		
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

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.

PROCESS_MOVE_ORDER

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_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

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_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 create 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	
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			

Parameter	Usage	Type	Required	Derived	Optional
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 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_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 create 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)
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 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.

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

See Also

Oracle Applications Message Reference Manual

This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Pick Release Application Program Interface

The Pick Release API is a public API that allows you to release a set of move order lines for pick wave move orders. This API creates move order line details for the lines that are released, and, depending on the parameters passed in, runs the pick confirm process immediately afterwards. The Pick Release API includes the public procedure pick_release.

Setting Up the Pick Release API

Parameter Descriptions

The following chart lists all parameters used by the pick_release procedure. Additional information on these parameters follows the chart.

PICK_RELEASE

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	Number			
x_msg_data	OUT	Varchar2			
p_mo_line_tbl	IN	PL/SQL Table	x		
p_auto_pick_confirm	IN	Number	x		
p_grouping_rule_id	IN	Number	x		
x_return_status	OUT	Varchar2			
x_msg_count	OUT	Varchar2			
x_msg_data	OUT	Varchar2			
x_pick_release_status	Out	PL/SQL Table			

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_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_mo_line_tbl

Indicates the PL/SQL table from which move order line records are chosen.

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.

Default Value: FND_API.G_MISS_NUM

p_grouping_rule_id

Overrides the organization level parameter and the move order header level grouping rule for generating pick slip numbers.

Default Value: FND_API.G_MISS_NUM

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.

x_pick_release_status

A table of records that specifies the pick release status for each move order line that is passed in.

Validation of Pick Release API

Standard Validation

Oracle Inventory validates all input parameters in the Pick_Release 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 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

See Also

Oracle Applications Message Reference Manual

This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

Pick Confirm Application Program Interface

The Pick Confirm 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.

Functional Overview

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

Parameter Descriptions

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			
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 following)

Parameter	Usage	Type	Required	Derived	Optional
p_mold_tbl	IN				x (you must indicate either this parameter or the preceding)
x_mmtt_tbl	OUT				
x_trolin_tbl	OUT				

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.

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

See Also

Oracle Applications Message Reference Manual

This manual is available in HTML format on the documentation CD-ROM for Release 11*i*.

7

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

This chapter contains information about the following Oracle Master Scheduling/MRP and Oracle supply Chain Planing open interfaces and application program interfaces:

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.

Table 7–1 Open Forecast Interface Table Description

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

Table 7–1 Open Forecast Interface Table Description

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	
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 to 2 (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:

Table 7–2 Open Master Schedule Interface Table Description

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

Table 7–2 Open Master Schedule Interface Table Description

Column Name	Type	Required	Derived	Optional
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	
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
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 11i.

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 to 2 (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:

Table 7–3 Oracle Master Scheduling/MRP Open Forecast Entries API

T_FORECAST_INTERFACE Column 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	
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
SOURCE_LINE_ID	Number			x
ATTRIBUTE1 - ATTRIBUTE15	Varchar2(150)			x
PROJECT ID	Number(15)			x
TASK ID	Number(15			x

Table 7–3 Oracle Master Scheduling/MRP Open Forecast Entries API

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

Open Forecast Interface Designator Table Description

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

Table 7–4 Oracle Master Scheduling/MRP Open Forecast Interface Designator

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
Success	5	Null
Failure	4	actual 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.

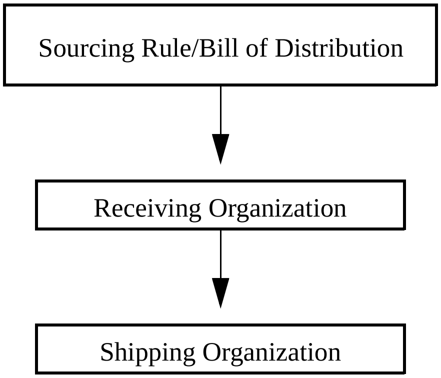
This API differs from the planning interfaces because it puts information directly into the Oracle Master Scheduling/MRP tables rather than inserting information into an interface table. You can process sourcing rule entries directly from within your calling module without running a concurrent process. 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:



- 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

Parameter	Usage	Type	Required	Derived	Optional
x_msg_count	OUT	Number			x
x_msg_data	OUT	Varchar2		x	
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	
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

Parameter	Type	Required	Derived	Optional
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
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	

Parameter	Type	Required	Derived	Optional
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	
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	

Parameter	Type	Required	Derived	Optional
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

Currently not 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.

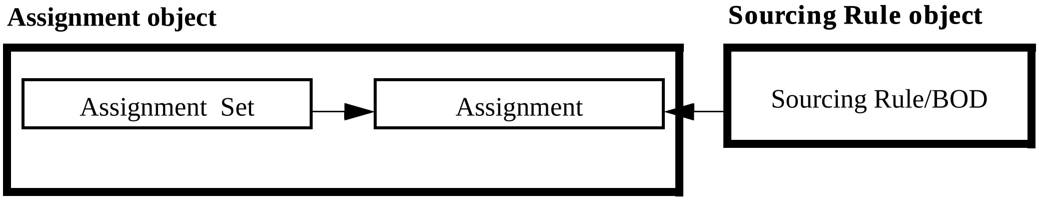
Condition	PROCESS_STATIS	ERROR_MESSAGE
Success	5	null
Failure	4	actual error message

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:



- 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—tem / 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

8

Oracle Purchasing Open Interfaces

This chapter contains information about the following Oracle Purchasing open interfaces:

- Requisitions Open Interface on page 8-2
- Purchasing Documents Open Interface on page 8-27
- Receiving Open Interface on page 8-70

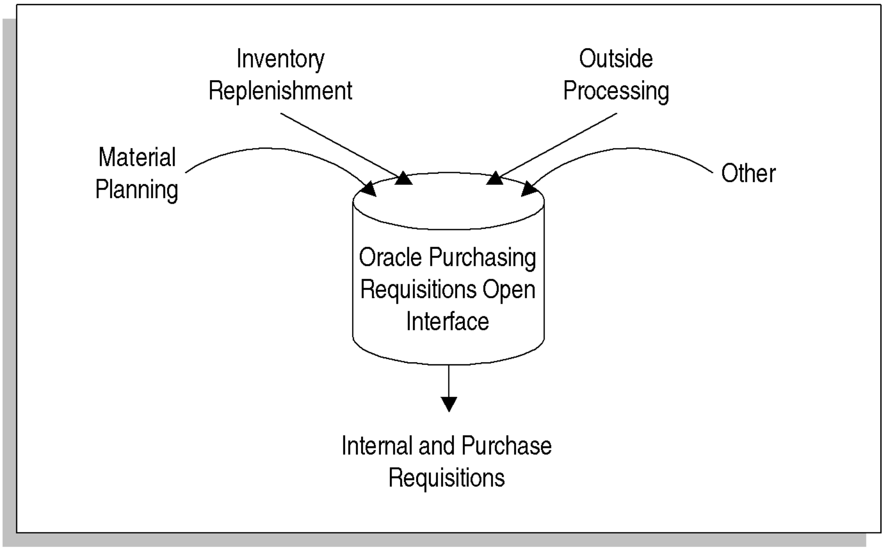
Requisitions Open Interface

You can automatically import requisitions from other Oracle Applications or your existing non-Oracle systems using the Requisitions Open Interface. This interface lets you integrate Oracle Purchasing quickly with new or existing applications such as material requirements planning, inventory management, and production control systems. Purchasing automatically validates your data and imports your requisitions. You can import requisitions as often as you want. Then, you can review these requisitions, approve or reserve funds for them if necessary, and place them on purchase orders or internal sales orders.

The purpose of this essay is to explain how to use the Requisitions Open Interface so that you can integrate other applications with Purchasing.

Functional Overview

Figure 8–1 Functional Overview


Inventory Replenishment
Outside Processing
Material Planning
Other
Oracle Purchasing Requisitions Open Interface
Internal and Purchase Requisitions

The diagram above shows the inputs and outputs that comprise the interface process.

You must write the program that inserts a single row into the PO_REQUISITIONS_INTERFACE_ALL and/or the PO_REQ_DIST_INTERFACE_ALL table for each requisition line that you want to import. Then you use the Submit Request window to launch the Requisition Import program for any set of rows. You identify the set of rows you want to import by setting the INTERFACE_SOURCE_CODE and BATCH_ID columns appropriately in the PO_REQUISITIONS_INTERFACE_ALL table. You then pass these values as parameters to the Requisition Import program. If you do not specify any values for these parameters, the program imports all the requisition lines in the PO_REQUISITIONS_INTERFACE_ALL table. You also specify the requisition grouping and numbering criteria as parameters to the Requisition Import program.

Each run of the Requisition Import program picks up distribution information from either the PO_REQUISITIONS_INTERFACE_ALL or the PO_REQ_DIST_INTERFACE_ALL table. The PO_REQ_DIST_INTERFACE_ALL table was used in Release 11, for Self-Service Purchasing (known then as Web Requisitions). In Self-Service Purchasing 4.0 and later, multiple distributions and the PO_REQ_DIST_INTERFACE_ALL table are not used, since Self-Service Purchasing updates the Purchasing interface tables directly rather than using Requisition Import. Therefore, in Release 11*i*, you should use the PO_REQ_DIST_INTERFACE_ALL table to create multiple distributions only for requisitions created in non-Oracle systems that use multiple distributions. As long as the Multiple Distributions field in the Requisition Import program is No (or blank), Requisition Import looks for distribution information in the PO_REQUISITIONS_INTERFACE_ALL table.

If MULTI_DISTRIBUTIONS is set to Y, the column REQ_DIST_SEQUENCE_ID in the PO_REQUISITIONS_INTERFACE_ALL table points to the primary key column, DIST_SEQUENCE_ID, in the PO_REQ_DIST_INTERFACE_ALL table to determine what distributions belong to which requisition line.

Note: If you import the requisitions from Oracle Master Scheduling/MRP, Oracle Order Management, or Oracle Inventory (INV), enter No for Multiple Distributions (set MULTI_DISTRIBUTIONS to N).

The Requisition Import program operates in three phases. In the first phase, the program validates your data and derives or defaults additional information. The program generates an error message for every validation that fails and creates a row in the PO_INTERFACE_ERRORS table with detailed information about each error. If the column MULTI_DISTRIBUTIONS in the PO_REQUISITIONS_INTERFACE_ALL table is Y, Requisition Import also checks for any records in the PO_

REQUISITIONS_INTERFACE_ALL table without corresponding distribution information in the PO_REQ_DIST_INTERFACE_ALL table and loads these as errors in the PO_INTERFACE_ERRORS table.

In the second phase, the program groups and numbers the validated requisition lines according to the following criteria. If you specify a value in the REQ_ NUMBER_SEGMENT1 column of the PO_REQUISITIONS_INTERFACE_ALL table, all lines with the same value for this column are grouped together under a requisition header. If you provide a value in the GROUP_CODE column, all lines with the same value in this column are grouped together under a requisition header. If you do not provide values in either of these columns, the Requisition Import program uses the Group By parameter to group lines together. If you do not provide a value for this parameter, the program uses the default Group By that you set up to group requisition lines. You can group requisition lines in one of the following ways that the Requisition Import program supports by:

- BUYER
- CATEGORY
- LOCATION
- VENDOR
- ITEM
- ALL (all requisition lines grouped under one header)

If you provide a value in the REQ_NUMBER_SEGMENT1 column of the PO_ REQUISITIONS_INTERFACE_ALL table, this value becomes the requisition number. If not, the Requisition Import program uses either the Last Requisition Number parameter if specified or the next unique number stored in the PO_ UNIQUE_IDENTIFIER_CONTROL table, adds 1 to this number, and starts numbering requisitions. If any of the requisition numbers generated already exists, the program loops until it finds a unique number. For every line that is successfully imported, a default distribution is created with the account information that you specify. (You specify account information in any of the following columns in either the PO_REQUISITIONS_INTERFACE_ALL or the PO_REQ_DIST_INTERFACE_ ALL table: CHARGE_ACCOUNT_ID, ACCRUAL_ACCOUNT_ID, VARIANCE_ ACCOUNT_ID, BUDGET_ACCOUNT_ID, or any of the CHARGE_ACCOUNT_ SEGMENT columns.) Requisition supply is also created for every approved requisition that is successfully imported.

In the third phase, the program deletes all the successfully processed rows in the interface tables, and creates a report which lists the number of interface records that were successfully imported and the number that were not imported. This report can be viewed by choosing View Output for the Requisition Import concurrent Request ID in the Requests window.

You can launch the Requisition Import Exceptions Report to view the rows that were not imported by the Requisition Import program along with the failure reason(s) for each row.

You can import approved or unapproved requisitions using the Requisitions Open Interface. If you are using requisition encumbrance, approved requisitions that you import automatically become pre-approved.

See Also

Requisition Import Process, *Oracle Purchasing User's Guide*

Requisition Import Exceptions Report, *Oracle Purchasing User's Guide*

Setting Up the Requisitions Interface

You must complete the following setup steps in Oracle Purchasing to use the Requisitions Open Interface. You must define a Requisition Import Group-By method in the Default region of the Purchasing Options window. For internally sourced requisitions, you must associate a customer with your deliver-to location using the Customer Addresses window.

All processing is initiated through standard report submission using the Submit Request window. The concurrent manager manages all processing, and as such it must be set up and running.

See Also

Defining Default Options, *Oracle Purchasing User's Guide*

Assigning a Business Purpose to a Customer Address, *Oracle Receivables User's Guide*

Inserting into the Requisitions Interface Tables

You load requisition lines from your source system or form into the requisitions interface table and/or the requisition distributions interface table. You insert one row for each requisition line that you want to import. You must provide values for all columns that are required. You may also have to provide values for columns that are conditionally required.

Requisitions Interface Table Description

The following graphic describes the requisitions interface table.

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL **Column Name**	**Type**	**Required**	**Derived**	**Optional**
TRANSACTION_ID	Number		x	
PROCESS_FLAG	Varchar2		x	
REQUEST_ID	Number		x	
PROGRAM_ID	Number		x	
PROGRAM_APPLICATION_ID	Number		x	
PROGRAM_UPDATE_DATE	Date		x	
LAST_UPDATED_BY	Number		x	
LAST_UPDATE_DATE	Date		x	
LAST_UPDATE_LOGIN	Number		x	
CREATION_DATE	Date		x	
CREATED_BY	Number		x	
INTERFACE_SOURCE_CODE	Varchar2	x		
INTERFACE_SOURCE_LINE_ID	Number			x
BATCH_ID	Number			x
GROUP_CODE	Varchar2			x
DELETE_ENABLED_FLAG	Varchar2	*No longer used*		
UPDATE_ENABLED_FLAG	Varchar2	*No longer used*		
SOURCE_TYPE_CODE	Varchar2	*conditionally*	*conditionally*	
REQUISITION_TYPE	Varchar2		x	
DESTINATION_TYPE_CODE	Varchar2	x		
AUTHORIZATION_STATUS	Varchar2	x		
PREPARER_ID	Number	x	*conditionally*	
PREPARER_NAME	Varchar2			x
APPROVER_ID	Number		x	
APPROVER_NAME	Varchar2			x
APPROVAL_PATH_ID	Number			x

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
REQUISITION_HEADER_ID	Number		x	
REQUISITION_LINE_ID	Number		x	
REQ_DISTRIBUTION_ID	Number		x	
REQ_NUMBER_SEGMENT1	Varchar2			x
REQ_NUMBER_SEGMENT2	Varchar2			x
REQ_NUMBER_SEGMENT3	Varchar2			x
REQ_NUMBER_SEGMENT4	Varchar2			x
REQ_NUMBER_SEGMENT5	Varchar2			x
HEADER_DESCRIPTION	Varchar2			x
HEADER_ATTRIBUTE_CATEGORY	Varchar2			x
HEADER_ATTRIBUTE1	Varchar2			x
HEADER_ATTRIBUTE2	Varchar2			x
HEADER_ATTRIBUTE3	Varchar2			x
HEADER_ATTRIBUTE4	Varchar2			x
HEADER_ATTRIBUTE5	Varchar2			x
HEADER_ATTRIBUTE6	Varchar2			x
HEADER_ATTRIBUTE7	Varchar2			x
HEADER_ATTRIBUTE8	Varchar2			x
HEADER_ATTRIBUTE9	Varchar2			x
HEADER_ATTRIBUTE10	Varchar2			x
HEADER_ATTRIBUTE11	Varchar2			x
HEADER_ATTRIBUTE12	Varchar2			x
HEADER_ATTRIBUTE13	Varchar2			x
HEADER_ATTRIBUTE14	Varchar2			x
HEADER_ATTRIBUTE15	Varchar2			x
URGENT_FLAG	Varchar2			x
RFQ_REQUIRED_FLAG	Varchar2			x
JUSTIFICATION	Varchar2			x

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL **Column Name**	**Type**	**Required**	**Derived**	**Optional**
NOTE_TO_BUYER	Varchar2			x
NOTE_TO_RECEIVER	Varchar2			x
NOTE_TO_APPROVER	Varchar2			x
ITEM_ID	Number	*conditionally*	*conditionally*	
ITEM_SEGMENT1	Varchar2			x
ITEM_SEGMENT2	Varchar2			x
ITEM_SEGMENT3	Varchar2			x
ITEM_SEGMENT4	Varchar2			x
ITEM_SEGMENT5	Varchar2			x
ITEM_SEGMENT6	Varchar2			x
ITEM_SEGMENT7	Varchar2			x
ITEM_SEGMENT8	Varchar2			x
ITEM_SEGMENT9	Varchar2			x
ITEM_SEGMENT10	Varchar2			x
ITEM_SEGMENT11	Varchar2			x
ITEM_SEGMENT12	Varchar2			x
ITEM_SEGMENT13	Varchar2			x
ITEM_SEGMENT14	Varchar2			x
ITEM_SEGMENT15	Varchar2			x
ITEM_SEGMENT16	Varchar2			x
ITEM_SEGMENT17	Varchar2			x
ITEM_SEGMENT18	Varchar2			x
ITEM_SEGMENT19	Varchar2			x
ITEM_SEGMENT20	Varchar2			x
ITEM_DESCRIPTION	Varchar2		x	
ITEM_REVISION	Varchar2			x
CATEGORY_ID	Number	*conditionally*	*conditionally*	
CATEGORY_SEGMENT1	Varchar2			x

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
CATEGORY_SEGMENT2	Varchar2			x
CATEGORY_SEGMENT3	Varchar2			x
CATEGORY_SEGMENT4	Varchar2			x
CATEGORY_SEGMENT5	Varchar2			x
CATEGORY_SEGMENT6	Varchar2			x
CATEGORY_SEGMENT7	Varchar2			x
CATEGORY_SEGMENT8	Varchar2			x
CATEGORY_SEGMENT9	Varchar2			x
CATEGORY_SEGMENT10	Varchar2			x
CATEGORY_SEGMENT11	Varchar2			x
CATEGORY_SEGMENT12	Varchar2			x
CATEGORY_SEGMENT13	Varchar2			x
CATEGORY_SEGMENT14	Varchar2			x
CATEGORY_SEGMENT15	Varchar2			x
CATEGORY_SEGMENT16	Varchar2			x
CATEGORY_SEGMENT17	Varchar2			x
CATEGORY_SEGMENT18	Varchar2			x
CATEGORY_SEGMENT19	Varchar2			x
CATEGORY_SEGMENT20	Varchar2			x
QUANTITY	Number	x		
UNIT_PRICE	Number		x	
CHARGE_ACCOUNT_ID	Number	x	*conditionally*	
CHARGE_ACCOUNT_SEGMENT1	Varchar2			x
CHARGE_ACCOUNT_SEGMENT2	Varchar2			x
CHARGE_ACCOUNT_SEGMENT3	Varchar2			x
CHARGE_ACCOUNT_SEGMENT4	Varchar2			x
CHARGE_ACCOUNT_SEGMENT5	Varchar2			x
CHARGE_ACCOUNT_SEGMENT6	Varchar2			x

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
CHARGE_ACCOUNT_SEGMENT7	Varchar2			x
CHARGE_ACCOUNT_SEGMENT8	Varchar2			x
CHARGE_ACCOUNT_SEGMENT9	Varchar2			x
CHARGE_ACCOUNT_SEGMENT10	Varchar2			x
CHARGE_ACCOUNT_SEGMENT11	Varchar2			x
CHARGE_ACCOUNT_SEGMENT12	Varchar2			x
CHARGE_ACCOUNT_SEGMENT13	Varchar2			x
CHARGE_ACCOUNT_SEGMENT14	Varchar2			x
CHARGE_ACCOUNT_SEGMENT15	Varchar2			x
CHARGE_ACCOUNT_SEGMENT16	Varchar2			x
CHARGE_ACCOUNT_SEGMENT17	Varchar2			x
CHARGE_ACCOUNT_SEGMENT18	Varchar2			x
CHARGE_ACCOUNT_SEGMENT19	Varchar2			x
CHARGE_ACCOUNT_SEGMENT20	Varchar2			x
CHARGE_ACCOUNT_SEGMENT21	Varchar2			x
CHARGE_ACCOUNT_SEGMENT22	Varchar2			x
CHARGE_ACCOUNT_SEGMENT23	Varchar2			x
CHARGE_ACCOUNT_SEGMENT24	Varchar2			x
CHARGE_ACCOUNT_SEGMENT25	Varchar2			x
CHARGE_ACCOUNT_SEGMENT26	Varchar2			x
CHARGE_ACCOUNT_SEGMENT27	Varchar2			x
CHARGE_ACCOUNT_SEGMENT28	Varchar2			x
CHARGE_ACCOUNT_SEGMENT29	Varchar2			x
CHARGE_ACCOUNT_SEGMENT30	Varchar2			x
ACCRUAL_ACCOUNT_ID	Number		x	
VARIANCE_ACCOUNT_ID	Number		x	
BUDGET_ACCOUNT_ID	Number		x	
UNIT_OF_MEASURE	Varchar2	*conditionally*	*conditionally*	

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL **Column Name**	**Type**	**Required**	**Derived**	**Optional**
UOM_CODE	Varchar2			x
LINE_TYPE_ID	Number		x	
LINE_TYPE	Varchar2			x
UN_NUMBER_ID	Number		*conditionally*	
UN_NUMBER	Varchar2			x
HAZARD_CLASS_ID	Number		*conditionally*	
HAZARD_CLASS	Varchar2			x
MUST_USE_SUGG_VENDOR_FLAG	Varchar2			x
REFERENCE_NUM	Varchar2			x
SOURCE_ORGANIZATION_ID	Number		*conditionally*	
SOURCE_ORGANIZATION_CODE	Varchar2			x
SOURCE_SUBINVENTORY	Varchar2			x
DESTINATION_ORGANIZATION_ID	Number	x	*conditionally*	
DESTINATION_ORGANIZATION_ CODE	Varchar2			x
DESTINATION_SUBINVENTORY	Varchar2	*conditionally*		
DELIVER_TO_LOCATION_ID	Number	x	*conditionally*	
DELIVER_TO_LOCATION_CODE	Varchar2			x
DELIVER_TO_REQUESTOR_ID	Number	x	*conditionally*	
DELIVER_TO_REQUESTOR_NAME	Varchar2			x
AUTOSOURCE_FLAG	Varchar2			x
AUTOSOURCE_DOC_HEADER_ID	Number		*conditionally*	
AUTOSOURCE_DOC_LINE_NUM	Number		*conditionally*	
DOCUMENT_TYPE_CODE	Varchar2		*conditionally*	
SUGGESTED_BUYER_ID	Number		*conditionally*	
SUGGESTED_BUYER_NAME	Varchar2			x
SUGGESTED_VENDOR_ID	Number		*conditionally*	
SUGGESTED_VENDOR_NAME	Varchar2			x

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
SUGGESTED_VENDOR_SITE_ID	Number		*conditionally*	
SUGGESTED_VENDOR_SITE	Varchar2			x
SUGGESTED_VENDOR_CONTACT_ID	Number		*conditionally*	
SUGGESTED_VENDOR_CONTACT	Varchar2		*conditionally*	
SUGGESTED_VENDOR_PHONE	Varchar2		*conditionally*	
SUGGESTED_VENDOR_ITEM_NUM	Varchar2			x
LINE_ATTRIBUTE_CATEGORY	Varchar2			x
LINE_ATTRIBUTE1	Varchar2			x
LINE_ATTRIBUTE2	Varchar2			x
LINE_ATTRIBUTE3	Varchar2			x
LINE_ATTRIBUTE4	Varchar2			x
LINE_ATTRIBUTE5	Varchar2			x
LINE_ATTRIBUTE6	Varchar2			x
LINE_ATTRIBUTE7	Varchar2			x
LINE_ATTRIBUTE8	Varchar2			x
LINE_ATTRIBUTE9	Varchar2			x
LINE_ATTRIBUTE10	Varchar2			x
LINE_ATTRIBUTE11	Varchar2			x
LINE_ATTRIBUTE12	Varchar2			x
LINE_ATTRIBUTE13	Varchar2			x
LINE_ATTRIBUTE14	Varchar2			x
LINE_ATTRIBUTE15	Varchar2			x
NEED_BY_DATE	Date	*conditionally*		
NOTE1_ID	Number		*conditionally*	
NOTE2_ID	Number		*conditionally*	
NOTE3_ID	Number		*conditionally*	
NOTE4_ID	Number		*conditionally*	
NOTE5_ID	Number		*conditionally*	

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL				
Column Name	**Type**	**Required**	**Derived**	**Optional**
NOTE6_ID	Number		*conditionally*	
NOTE7_ID	Number		*conditionally*	
NOTE8_ID	Number		*conditionally*	
NOTE9_ID	Number		*conditionally*	
NOTE10_ID	Number		*conditionally*	
NOTE1_TITLE	Varchar2			x
NOTE2_TITLE	Varchar2			x
NOTE3_TITLE	Varchar2			x
NOTE4_TITLE	Varchar2			x
NOTE5_TITLE	Varchar2			x
NOTE6_TITLE	Varchar2			x
NOTE7_TITLE	Varchar2			x
NOTE8_TITLE	Varchar2			x
NOTE9_TITLE	Varchar2			x
NOTE10_TITLE	Varchar2			x
DIST_ATTRIBUTE_CATEGORY	Varchar2			x
DISTRIBUTION_ATTRIBUTE1	Varchar2			x
DISTRIBUTION_ATTRIBUTE2	Varchar2			x
DISTRIBUTION_ATTRIBUTE3	Varchar2			x
DISTRIBUTION_ATTRIBUTE4	Varchar2			x
DISTRIBUTION_ATTRIBUTE5	Varchar2			x
DISTRIBUTION_ATTRIBUTE6	Varchar2			x
DISTRIBUTION_ATTRIBUTE7	Varchar2			x
DISTRIBUTION_ATTRIBUTE8	Varchar2			x
DISTRIBUTION_ATTRIBUTE9	Varchar2			x
DISTRIBUTION_ATTRIBUTE10	Varchar2			x
DISTRIBUTION_ATTRIBUTE11	Varchar2			x
DISTRIBUTION_ATTRIBUTE12	Varchar2			x

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL **Column Name**	**Type**	**Required**	**Derived**	**Optional**
DISTRIBUTION_ATTRIBUTE13	Varchar2			x
DISTRIBUTION_ATTRIBUTE14	Varchar2			x
DISTRIBUTION_ATTRIBUTE15	Varchar2			x
GOVERNMENT_CONTEXT	Varchar2			x
GL_DATE	Date		*conditionally*	
USSGL_TRANSACTION_CODE	Varchar2			x
PREVENT_ENCUMBRANCE_FLAG	Varchar2		x	
CURRENCY_CODE	Varchar2			x
CURRENCY_UNIT_PRICE	Number		*conditionally*	
RATE	Number		*conditionally*	
RATE_DATE	Date	*conditionally*		
RATE_TYPE	Varchar2	*conditionally*		
WIP_ENTITY_ID	Number	*conditionally*		
WIP_LINE_ID	Number			x
WIP_OPERATION_SEQ_NUM	Number			x
WIP_RESOURCE_SEQ_NUM	Number			x
WIP_REPETITIVE_SCHEDULE_ID	Number	*conditionally*		
BOM_RESOURCE_ID	Number	*conditionally*		
EXPENDITURE_ORGANIZATION_ID	Number	*conditionally*		
EXPENDITURE_TYPE	Varchar2	*conditionally*		
PROJECT_ACCOUNTING_CONTEXT	Varchar2			x
PROJECT_ID	Number	*conditionally*	*conditionally*	
PROJECT_NUM	Varchar2			x
TASK_ID	Number	*conditionally*	*conditionally*	
TASK_NUM	Varchar2			x
EXPENDITURE_ITEM_DATE	Date			x
TRANSACTION_REASON_CODE	Varchar2			x
ORG_ID	Number	*conditionally*		

Table 8–1 Requisitions Open Interface (Requisitions)

PO_REQUISITIONS_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
ALLOCATION_TYPE	Varchar2			x
ALLOCATION_VALUE	Number			x
MULTI_DISTRIBUTIONS	Varchar2			x
REQ_DIST_SEQUENCE_ID	Number			x
KANBAN_CARD_ID	Number			x
EMERGENCY_PO_NUM	Varchar2	*No longer used*		
AWARD_ID	Number			
TAX_NAME	Varchar2	*No longer used*		
END_ITEM_UNIT_NUMBER	Varchar2			x
TAX_CODE_ID	Number			x

Requisition Distributions Interface Table Description

The following graphic describes the requisition distributions interface table. This table was used in Release 11 to create multiple distributions for Self-Service Purchasing requisitions but is no longer used by Self-Service Purchasing in Release 11*i*. The table remains in case you need to import multiple-distribution requisitions from non-Oracle systems.

Table 8–2 Requisitions Open Interface (Distributions)

PO_REQ_DIST_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
PROJECT_ACCOUNTING_CONTEXT	Varchar2			x
EXPENDITURE_ORGANIZATION_ID	Number	*conditionally*		
PROJECT_ID	Number	*conditionally*	*conditionally*	
TASK_ID	Number	*conditionally*	*conditionally*	
EXPENDITURE_ITEM_DATE	Date			x
GL_DATE	Date		*conditionally*	
DIST_ATTRIBUTE_CATEGORY	Varchar2			x
DISTRIBUTION_ATTRIBUTE1	Varchar2			x
DISTRIBUTION_ATTRIBUTE2	Varchar2			x

Table 8–2 Requisitions Open Interface (Distributions)

PO_REQ_DIST_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
DISTRIBUTION_ATTRIBUTE3	Varchar2			x
DISTRIBUTION_ATTRIBUTE4	Varchar2			x
DISTRIBUTION_ATTRIBUTE5	Varchar2			x
DISTRIBUTION_ATTRIBUTE6	Varchar2			x
DISTRIBUTION_ATTRIBUTE7	Varchar2			x
DISTRIBUTION_ATTRIBUTE8	Varchar2			x
DISTRIBUTION_ATTRIBUTE9	Varchar2			x
DISTRIBUTION_ATTRIBUTE10	Varchar2			x
DISTRIBUTION_ATTRIBUTE11	Varchar2			x
DISTRIBUTION_ATTRIBUTE12	Varchar2			x
DISTRIBUTION_ATTRIBUTE13	Varchar2			x
DISTRIBUTION_ATTRIBUTE14	Varchar2			x
DISTRIBUTION_ATTRIBUTE15	Varchar2			x
CHARGE_ACCOUNT_ID	Number	x	*conditionally*	
CHARGE_ACCOUNT_SEGMENT1	Varchar2			x
CHARGE_ACCOUNT_SEGMENT2	Varchar2			x
CHARGE_ACCOUNT_SEGMENT3	Varchar2			x
CHARGE_ACCOUNT_SEGMENT4	Varchar2			x
CHARGE_ACCOUNT_SEGMENT5	Varchar2			x
CHARGE_ACCOUNT_SEGMENT6	Varchar2			x
CHARGE_ACCOUNT_SEGMENT7	Varchar2			x
CHARGE_ACCOUNT_SEGMENT8	Varchar2			x
CHARGE_ACCOUNT_SEGMENT9	Varchar2			x
CHARGE_ACCOUNT_SEGMENT10	Varchar2			x
CHARGE_ACCOUNT_SEGMENT11	Varchar2			x
CHARGE_ACCOUNT_SEGMENT12	Varchar2			x
CHARGE_ACCOUNT_SEGMENT13	Varchar2			x
CHARGE_ACCOUNT_SEGMENT14	Varchar2			x

Table 8–2 Requisitions Open Interface (Distributions)

PO_REQ_DIST_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
CHARGE_ACCOUNT_SEGMENT15	Varchar2			x
CHARGE_ACCOUNT_SEGMENT16	Varchar2			x
CHARGE_ACCOUNT_SEGMENT17	Varchar2			x
CHARGE_ACCOUNT_SEGMENT18	Varchar2			x
CHARGE_ACCOUNT_SEGMENT19	Varchar2			x
CHARGE_ACCOUNT_SEGMENT20	Varchar2			x
CHARGE_ACCOUNT_SEGMENT21	Varchar2			x
CHARGE_ACCOUNT_SEGMENT22	Varchar2			x
CHARGE_ACCOUNT_SEGMENT23	Varchar2			x
CHARGE_ACCOUNT_SEGMENT24	Varchar2			x
CHARGE_ACCOUNT_SEGMENT25	Varchar2			x
CHARGE_ACCOUNT_SEGMENT26	Varchar2			x
CHARGE_ACCOUNT_SEGMENT27	Varchar2			x
CHARGE_ACCOUNT_SEGMENT28	Varchar2			x
CHARGE_ACCOUNT_SEGMENT29	Varchar2			x
CHARGE_ACCOUNT_SEGMENT30	Varchar2			x
GROUP_CODE	Varchar2			x
PROJECT_NUM	Varchar2			x
TASK_NUM	Varchar2			x
EXPENDITURE_TYPE	Varchar2	*conditionally*		
DIST_SEQUENCE_ID	Number	*conditionally*		
ALLOCATION_TYPE	Varchar2	*conditionally*		
ALLOCATION_VALUE	Number	*conditionally*		
BATCH_ID	Number			x
DISTRIBUTION_NUMBER	Number	x		
ITEM_ID	Number	*conditionally*	*conditionally*	
ACCRUAL_ACCOUNT_ID	Number		x	
VARIANCE_ACCOUNT_ID	Number		x	

Table 8–2 Requisitions Open Interface (Distributions)

PO_REQ_DIST_INTERFACE_ALL Column Name	Type	Required	Derived	Optional
BUDGET_ACCOUNT_ID	Number		x	
USSGL_TRANSACTION_CODE	Varchar2			x
GOVERNMENT_CONTEXT	Varchar2			x
PREVENT_ENCUMBRANCE_FLAG	Varchar2			x
TRANSACTION_ID	Number		x	
PROCESS_FLAG	Varchar2		x	
REQUEST_ID	Number		x	
PROGRAM_ID	Number		x	
PROGRAM_APPLICATION_ID	Number		x	
PROGRAM_UPDATE_DATE	Date		x	
LAST_UPDATED_BY	Number		x	
LAST_UPDATE_DATE	Date		x	
LAST_UPDATE_LOGIN	Number		x	
CREATION_DATE	Date			x
DESTINATION_ORGANIZATION_ID	Number	x	*conditionally*	
DESTINATION_SUBINVENTORY	Varchar2	*conditionally*		
DESTINATION_TYPE_CODE	Varchar2	x		
CREATED_BY	Number			x
INTERFACE_SOURCE_CODE	Varchar2	x		
INTERFACE_SOURCE_LINE_ID	Number			x
REQUISITION_HEADER_ID	Number		x	
REQUISITION_LINE_ID	Number		x	
REQ_DISTRIBUTION_ID	Number		x	
REQ_NUMBER_SEGMENT1	Varchar2			x
QUANTITY	Number	*conditionally*	*conditionally*	
ORG_ID	Number	*conditionally*		
UPDATE_ENABLED_FLAG	Varchar2	*No longer used*		

Required Data

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

- INTERFACE_SOURCE_CODE to identify the source of your imported requisitions
- DESTINATION_TYPE_CODE
- AUTHORIZATION_STATUS
- PREPARER_ID or PREPARER_NAME
- QUANTITY
- CHARGE_ACCOUNT_ID or charge account segment values
- DESTINATION_ORGANIZATION_ID or DESTINATION_ORGANIZATION_CODE
- DELIVER_TO_LOCATION_ID or DELIVER_TO_LOCATION_CODE
- DELIVER_TO_REQUESTOR_ID or DELIVER_TO_REQUESTOR_NAME

You must always enter values for the following required columns if you load rows into the PO_REQ_DIST_INTERFACE_ALL table:

- CHARGE_ACCOUNT_ID or charge account segment values
- DISTRIBUTION_NUMBER (Although Requisition Import won't give an error if you don't provide a value, a DISTRIBUTION_NUMBER makes it easier to query multiple distributions in the Distributions windows in Purchasing.)
- DESTINATION_ORGANIZATION_ID
- DESTINATION_TYPE_CODE
- INTERFACE_SOURCE_CODE

Additionally, you may have to enter values for the following conditionally required columns.

In the PO_REQUISITIONS_INTERFACE_ALL table:

- You must provide a SOURCE_TYPE_CODE if the DESTINATION_TYPE_CODE is 'EXPENSE' or 'SHOP FLOOR'. You must provide an ITEM_ID or item segment values if the SOURCE_TYPE_CODE or DESTINATION_TYPE_CODE is 'INVENTORY'.

- For one-time items and amount-based line types, you must provide a CATEGORY_ID or category segment values. You must additionally provide a UNIT_OF_MEASURE or UOM_CODE for one-time items. For MRP or Inventory planned items, you must also provide a NEED_BY_DATE.
- You must provide the RATE_DATE and RATE_TYPE if you provide a value in the CURRENCY_CODE column.
- If you are using Oracle Work in Process and the DESTINATION_TYPE_CODE is 'SHOP FLOOR', you must provide values for the following columns:

  WIP_ENTITY_ID

  BOM_RESOURCE_ID

  WIP_REPETITIVE_SCHEDULE_ID, if the entity is a repetitive schedule
- ITEM_ID may also be required. See: Validation on page 8-24.

In the PO_REQ_DIST_INTERFACE_ALL table:

- You must provide a DIST_SEQUENCE_ID if MULTI_DISTRIBUTIONS is set to Y.
- If you do not enter a value in the QUANTITY column, you must enter values in the ALLOCATION_TYPE and ALLOCATION_VALUE columns.

In both the PO_REQUISITIONS_INTERFACE_ALL and PO_REQ_DIST_INTERFACE_ALL tables:

- You must provide an ORG_ID if you have a Multiple Organization Support setup.
- If you are using Oracle Projects and the PROJECT_ACCOUNTING_CONTEXT is 'Y', you must enter the relevant project accounting information in the following columns:

  PROJECT_NUM or PROJECT_ID

  TASK_NUM or TASK_ID

  EXPENDITURE_TYPE

  EXPENDITURE_ORGANIZATION_ID

  If Oracle Project Manufacturing is installed, Project Reference Enabled is selected in the Project Manufacturing Organization Parameters window, and the PROJECT_ACCOUNTING_CONTEXT is 'Y', you must enter the relevant project information in the following columns:

PROJECT_NUM or PROJECT_ID

TASK_NUM or TASK_ID, if the destination type is Inventory or Shop Floor

If you are creating multiple distributions, project information must be entered in the PO_REQ_DIST_INTERFACE_ALL table.

- You must provide a DESTINATION_SUBINVENTORY if the DESTINATION_TYPE_CODE is 'INVENTORY'.

For additional information on conditionally required columns, see: Validation on page 8-24.

Derived Data

The Requisition Import program derives or defaults the columns identified as derived using logic similar to that used by the Requisitions window. Oracle Purchasing never overrides information that you provide in derived columns. (Supplier sourcing is an exception to this rule). Column pairs like APPROVER_ID / APPROVER_NAME, NOTE_ID / NOTE_TITLE, and DESTINATION_ORGANIZATION_ID / DESTINATION_ORGANIZATION_CODE in the requisitions interface table allow you to enter the user-displayed value in the interface table and the program derives the associated unique identifier. If there is a conflict between the two values, the identifier overrides the user-displayed value.

In the PO_REQUISITIONS_INTERFACE_ALL table:

- For interface lines with a DESTINATION_TYPE_CODE of 'INVENTORY', the program derives the SOURCE_TYPE_CODE. The REQUISITION_TYPE is derived from the SOURCE_TYPE_CODE.
- The Requisition Import program automatically derives sourcing information for both your inventory and purchase requisition lines if you set the AUTOSOURCE_FLAG to 'Y' and set up the sourcing rules for the item. For inventory-sourced requisition lines, the program derives the following columns:

  SOURCE_ORGANIZATION_ID

  SOURCE_SUBINVENTORY

  For supplier-sourced requisition lines, the program derives the following columns:

  SUGGESTED_VENDOR_ID

  SUGGESTED_VENDOR_SITE_ID

  SUGGESTED_VENDOR_CONTACT_ID

SUGGESTED_BUYER_ID

AUTOSOURCE_DOC_HEADER_ID

AUTOSOURCE_DOC_LINE_NUM

DOCUMENT_TYPE_CODE

- If you set the AUTOSOURCE_FLAG to 'P' (for Partially required) and set up the sourcing rules for the item, the program derives the following columns for inventory-sourced requisition lines:

  SOURCE_ORGANIZATION_ID

  SOURCE_SUBINVENTORY

  If you set the AUTOSOURCE_FLAG to 'P' and set up the sourcing rules for the item, the program uses the following columns for supplier-sourced requisition lines, if they're provided in the requisitions interface table:

  SUGGESTED_VENDOR_ID

  SUGGESTED_VENDOR_SITE_ID

  If the above columns are not provided when the AUTOSOURCE_FLAG is set to 'P', the program derives them from the sourcing rules.

- If you set the AUTOSOURCE_FLAG to 'P' and set up the sourcing rules for the item, the program derives the following columns for supplier-sourced requisition lines:

  SUGGESTED_VENDOR_CONTACT_ID

  SUGGESTED_BUYER_ID

  AUTOSOURCE_DOCUMENT_HEADER_ID

  AUTOSOURCE_DOCUMENT_LINE_NUM

  DOCUMENT_TYPE_CODE

- Item pricing information is also derived in the UNIT_PRICE and CURRENCY_UNIT_PRICE columns. If no sourcing rules are found for the item, supplier sourcing fails and the UNIT_PRICE is defaulted from the item master for supplier requisition lines and from the CST_ITEM_COSTS_FOR_GL_VIEW for internal requisitions.

In the PO_REQ_DIST_INTERFACE_ALL table:

- The Requisition Import program derives the QUANTITY (if a QUANTITY is not indicated) if ALLOCATION_TYPE and ALLOCATION_VALUE are provided.

In both the PO_REQUISITIONS_INTERFACE_ALL and PO_REQ_DIST_INTERFACE_ALL tables:

- You can provide the segment values for the item, category, and charge account. The Requisition Import program derives the ITEM_ID and CATEGORY_ID from the requisitions interface table and the CHARGE_ACCOUNT_ID from either the requisitions interface table or the requisition distributions interface table. In both the requisitions and requisition distributions interface tables, the ACCRUAL_ACCOUNT_ID, BUDGET_ACCOUNT_ID, and VARIANCE_ACCOUNT_ID are derived based on the DESTINATION_TYPE_CODE.
- The following columns are control columns that the Requisition Import program derives to provide audit trail and relational integrity throughout the interface process:

  CREATION_DATE

  CREATED_BY

  LAST_UPDATE_DATE

  LAST_UPDATED_BY

  LAST_UPDATE_LOGIN

  PROGRAM_ID

  PROGRAM_APPLICATION_ID

  PROGRAM_UPDATE_DATE

  REQUEST_ID

Optional Data

You can enter header, line, and distribution-level descriptive flexfield information in the interface tables. You can enter up to ten notes for each requisition that you import. The Requisitions Open Interface also lets you enter foreign currency information, project accounting information, UN number, and hazard class information. You can enter the justification for the requisition and indicate whether the requisition is urgent. You can also provide item revision, source, and destination subinventory information. If you are using requisition encumbrance, you can also provide a USSGL transaction code.

Validation

Standard Validation

Oracle Purchasing validates all required columns in the interface table. For specific information on the data implied by these columns, see the *Oracle Purchasing Technical Reference Manual, Release 11i*, for details.

Other Validation

Purchasing also performs the following cross validations. If a row in the interface tables fails validation for any reason, the program sets the PROCESS_FLAG in the interface table to 'ERROR' and enters details about every error on that row into the PO_INTERFACE_ERRORS table.

If you enter a SOURCE_TYPE_CODE of 'INVENTORY', the ITEM_ID is required and the item must be stock-enabled for the source organization and internal-order-enabled for the purchasing and destination organizations. The DELIVER_TO_LOCATION_ID must be valid for the destination organization and a customer must be associated with that location in Purchasing. If you also enter a SOURCE_SUBINVENTORY, the item must either be valid in the subinventory, or it must not be restricted to a subinventory. For MRP-sourced internal requisitions, the SOURCE_SUBINVENTORY must be a non-nettable subinventory for intra-organization transfers.

If you enter a SOURCE_TYPE_CODE of 'VENDOR' and provide an ITEM_ID, the item must be purchasing-enabled for the purchasing and destination organizations.

If you enter a DESTINATION_TYPE_CODE of 'INVENTORY', the ITEM_ID is required and it must be stock-enabled for the destination organization. If you also enter a DESTINATION_SUBINVENTORY, the item must either be valid in the subinventory or it must not be restricted to a subinventory.

If you enter a DESTINATION_TYPE_CODE of 'SHOP FLOOR', the ITEM_ID is required, and it must be an outside-operation item and purchasing-enabled for the purchasing and destination organizations. The LINE_TYPE_ID must be an outside-operation line type as well.

If you provide a CURRENCY_CODE, the RATE, RATE_DATE, and RATE_TYPE must be provided.

If you are using requisition encumbrance, the GL_DATE that you enter must be in an open or future General Ledger period and an open Purchasing period. Furthermore, if you are using Oracle Inventory, the GL_DATE must be in an open Inventory period for inventory-sourced requisitions.

Resolving Failed Requisitions Interface Rows

Error Messages

Oracle Purchasing may display specific error messages during interface processing. For more details on these messages, please see the *Oracle Applications Message Reference Manual,* in HTML format on the documentation CD-ROM for Release 11*i*.

Viewing Failed Transactions

You can report on all rows that failed validation by using the Requisition Import Exceptions report. For every transaction in the interface table that fails validation, the Requisition Import Exceptions report lists all the columns that failed validation along with the reason for the failure.

You can identify failed transactions in the requisitions interface tables by selecting rows with a PROCESS_FLAG of 'ERROR'. For any previously processed set of rows identified by INTERFACE_SOURCE_CODE and BATCH_ID, only rows that failed validation remain in the interface table, as all the successfully imported rows are deleted.

For each row in the requisitions interface or requisition distributions interface table that fails validation, the Requisition Import program creates one or more rows with error information in the PO_INTERFACE_ERRORS table.

Rescheduling Requisitions

If you use Oracle Master Scheduling / MRP or a non-Oracle MRP system with Oracle Purchasing, you may find that you need to reschedule requisitions as your planning requirements change. Purchasing's Requisition Import program lets you reschedule requisition lines according to changes in your planned orders.

Reschedule Interface Table

You can reschedule requisitions from your planning application with the Reschedule Interface table. Since you have already loaded your requisitions into Purchasing, you simply need to identify for Purchasing the requisition lines you want to reschedule. After you identify each line to reschedule, you can update the quantity and the need-by date for the corresponding requisition line. You decide when to import the information from the requisitions interface table into Purchasing. Purchasing lets you use the Reschedule Interface table as often as you want.

Understanding the PO_RESCHEDULE_INTERFACE Table

PO_RESCHEDULE_INTERFACE is the table Purchasing uses to import information for requisition lines your planning system has rescheduled. One row in the table corresponds to a requisition line whose quantity or need-by date you want to change. Requisition Import updates your requisition lines within Purchasing with the information in this table. The table PO_RESCHEDULE_INTERFACE consists of columns Purchasing uses to identify requisition lines for update. The table PO_RESCHEDULE_INTERFACE contains the following columns:

Table 8–3 PO_RESCHEDULE_INTERFACE

Column Name	Null?	Type
LINE_ID	NOT NULL	NUMBER
QUANTITY		NUMBER
NEED_BY_DATE		DATE
PROCESS_ID		NUMBER
LAST_UPDATE_DATE		DATE
LAST_UPDATED_BY		NUMBER
LAST_UPDATE_LOGIN		NUMBER
CREATION_DATE		DATE
CREATED_BY		NUMBER
REQUEST_ID		NUMBER
PROGRAM_APPLICATION_ID		NUMBER
PROGRAM_ID		NUMBER
PROGRAM_UPDATE_DATE		DATE

The columns listed below are foreign keys to the following tables and columns:

Table 8–4 Foreign Keys

Foreign Key	Table	Column
LINE_ID	PO_REQUISITION_LINES	LINE_ID
QUANTITY	PO_REQUISITION_LINES	QUANTITY
NEED_BY_DATE	PO_REQUISITION_LINES	NEED_BY_DATE

The column LINE_ID identifies a requisition line which your planning system reschedules. The columns QUANTITY and NEED_BY_DATE contain new information for the requisition lines your planning system updates.

The other columns in the table store the same information the PO_REQUISITIONS_INTERFACE_ALL table uses to track when you place data in the PO_RESCHEDULE_INTERFACE table.

Columns Reserved for Requisition Import

Requisition Import inserts values into the column PROCESS_ID. Requisition Import inserts the PROCESS_ID to identify all requisition lines which you reschedule at one time. You should not insert any data in this column.

Purchasing Documents Open Interface

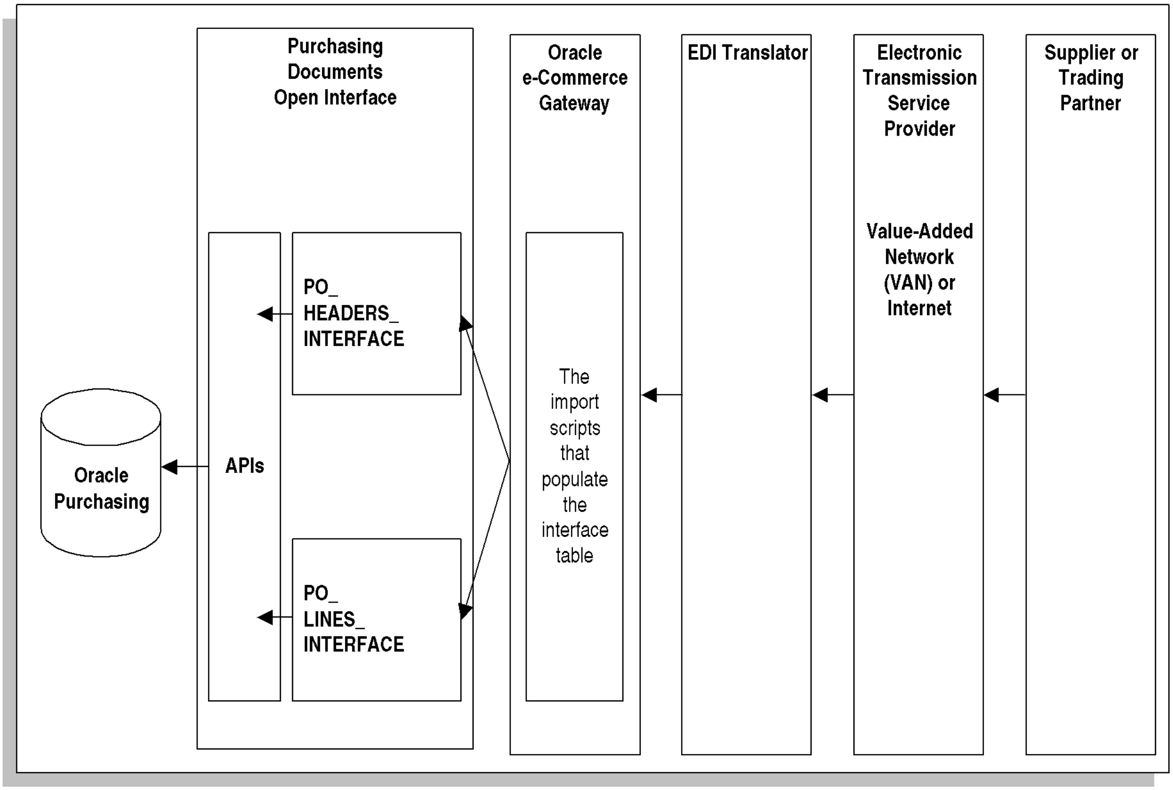
You can automatically import and update price/sales catalog information and request for quotation (RFQ) responses from suppliers through the Purchasing Documents Open Interface. The Purchasing Documents Open Interface uses Application Program Interfaces (APIs) to process catalog data in the Oracle Applications interface tables to ensure that it is valid before importing it into Oracle Purchasing. After validating the price/sales catalog information or RFQ responses, the Purchasing Documents Open Interface program converts the information, including price break information, in the interface tables into blanket purchase agreements or catalog quotations in Purchasing.

You can choose whether to import the data as blanket purchase agreements or catalog quotations. You can also choose to update your item master and apply sourcing rules and release generation methods to the imported item for both blanket purchase agreements and quotations. Blanket purchase agreements or quotations can also be replaced with the latest price/sales catalog information when your supplier sends a replacement catalog, or updated when the supplier sends an updated catalog.

One way to import price/sales catalog data is through Electronic Data Interchange (EDI). The Purchasing Documents Open Interface supports the EDI transmissions of the price/sales catalogs (ANSI X12 832 or EDIFACT PRICAT) and responses to RFQs (ANSI X12 843 or EDIFACT QUOTES).

Functional Overview

Figure 8–2 Functional Overview


Purchasing Documents Open Interface
Oracle e-Commerce Gateway
EDI Translator
Electronic Transmission Service Provider
Supplier or Trading Partner
Value-Added Network (VAN) or Internet
PO_ HEADERS_ INTERFACE
APIs
PO_ LINES_ INTERFACE
Oracle Purchasing
The import scripts that populate the interface table

The figure above shows the flow of price/sales catalog information from the supplier or trading partner, to Oracle e-Commerce Gateway, to the Purchasing Documents Open Interface, and finally into Purchasing. The PO_HEADERS_ INTERFACE and PO_LINES_INTERFACE tables can also be loaded manually, through a program you write.

If you load the interface tables through e-Commerce Gateway, then the supplier must provide the price/sales catalog information as a flat file using an EDI translator according to the EDI interface file definitions. Then, the EDI Catalog Inbound program (or the EDI Response to RFQ Inbound program) loads the information into the PO_HEADERS_INTERFACE table and the PO_LINES_INTERFACE table, which includes line, shipment, and price break information.

In the Parameters window of the EDI Catalog Inbound program (or EDI Response to RFQ Inbound program), you specify the name of the flat file and designate how the data sent by the supplier is to be used—if blanket purchase agreements or catalog quotations are to be created, if items are created or updated in the item master, if sourcing rules are created or updated. (You also specify the location of the flat file through an e-Commerce Gateway profile option. See: *Oracle e-Commerce Gateway User's Guide*.)

The EDI inbound program and the Purchasing Documents Open Interface program are run as a request set when you choose Submit Request in the EDI import programs window. The EDI inbound program loads the interface tables; the Purchasing Documents Open Interface program validates the data and loads the validated data into Purchasing. You can also run the Purchasing Documents Open Interface program separately in the Submit Request window in Purchasing, after the data is loaded into the interface tables.

You can view the status of your submission by making note of the Request ID number and selecting View My Requests from the Help menu.

The Purchasing Documents Open Interface program receives the data, derives and defaults any missing data, and validates the data. If no errors are found in the submission process, the data in the Purchasing Documents Open Interface tables is loaded into the PO_HEADERS, PO_LINES, and PO_LINE_LOCATIONS tables in Purchasing to create the blanket purchase agreement or catalog quotation, including price breaks if any. The creation of items (if you allow updating of the item master) or sourcing rules also populates the corresponding tables (such as MTL_SYSTEM_ITEMS, ASL_ITEMS, ASL_SUPPLIERS, and ASL_DOCUMENTS).

If the Purchasing Documents Open Interface program finds errors in the interface tables, such as incomplete information from the supplier, the record identification number and the details of the error are written to the PO_INTERFACE_ERRORS table. You can launch the Purchasing Interface Errors Report in Purchasing to view the rows that were not imported by the Purchasing Documents Open Interface along with the failure reason(s) for each row.

Record and Error Processing

To detect errors, the Purchasing Documents Open Interface program first processes a record from the PO_HEADERS_INTERFACE table. Then, the program processes the child records in the PO_LINES_INTERFACE table before going on to the next PO_HEADERS_INTERFACE record.

If the program gets an error while processing a record, the program writes the error details to the PO_INTERFACE_ERRORS table and increments the record's error counter. It then flags the record as "Not Processable."

The Purchasing Documents Open Interface saves or errors out on a line-by-line basis. This means that if an error is found in a document line, only that line is rolled back (not submitted to Purchasing), and you can find the error in the PO_INTERFACE_ERRORS table. Because the Purchasing Documents Open Interface saves or errors out line by line, it can accept partial documents.

If an error is found in a header, none of its lines are processed. The Purchasing Documents Open Interface rolls back the header, does not process its lines, and does the following:

- Sets the PROCESS_CODE column value to REJECTED in the PO_HEADERS_INTERFACE table.
- Writes out the record identification number and the details of the error to the PO_INTERFACE_ERRORS table.
- Begins processing the next header record.

If no processing errors are found during processing, the header record and all successfully submitted child records are loaded into Purchasing, and then flagged as processed by setting the PROCESS_CODE column to ACCEPTED.

When the supplier sends an updated price/sales catalog, the Purchasing Documents Open Interface sets the PROCESS_CODE column to NOTIFIED for those lines with prices that exceed your price tolerance. For these price updates only, the Purchasing Documents Open Interface waits for you to accept or reject the price increase in the Exceeded Price Tolerances window before completing or rejecting the price change.

Attention: It is important to check the Purchasing Interface Errors report (a report of the errors in the PO_INTERFACE_ ERRORS table) after you import the price/sales catalog. Because the Purchasing Documents Open Interface saves or errors out line by line, it can accept partial documents. Therefore, to see which document lines were not submitted because of errors, you must check the Purchasing Interface Errors report.

Original, Replace, and Update Submissions

If you are using e-Commerce Gateway to import the data (in the form of flat files) into the Purchasing Documents Open Interface, your supplier can send you a flat file with one of three action codes (in the ACTION column of the PO_HEADERS_ INTERFACE table): Original, Replace, or Update. If you're not using e-Commerce Gateway, your import program needs to specify the action code.

Your supplier should use the Original action code for a new price/sales catalog submission, and Replace or Update action codes for subsequent submissions.

Original

A file with an action code of Original is one in which all the price/sales catalog information is new to your system.

Choose Original when you're submitting price/sales catalog information for the first time.

For an Original submission, the Purchasing Documents Open Interface first checks if a document in the submission already exists in Purchasing. It checks for a document with the same supplier document number (VENDOR_DOC_NUM) and supplier (VENDOR_ID or VENDOR_NAME) that is not finally closed or canceled. If an active, matching document already exists, the document in the Original submission is not created and an error is logged in the Purchasing Interface Errors report.

Replace

A file with an action code of Replace replaces already-created blanket purchase agreements or catalog quotations with new documents containing the new price/sales catalog information. The Purchasing Documents Open Interface replaces these documents by doing the following:

- First, it looks for documents that have the same document type (DOCUMENT_TYPE_CODE), supplier (VENDOR_ID or VENDOR_NAME), and supplier document number (VENDOR_DOC_NUM) as the replacement documents. (A supplier document number is a field that is specified in the flat file that the supplier sends.)
- Next, among the matching documents, it looks for documents with effectivity dates that are the same as, or within the effectivity dates of, the replacement documents.
- Then, it invalidates the old documents by setting their expiration dates to START_DATE -1 (the start date, minus one day) and creates new documents with the new price/sales catalog information, within the original effectivity dates.

Choose Replace when you want to replace the whole document because most of its fields, header, and line information have changed or are out of date.

As with an Original submission, the Purchasing Documents Open Interface checks that there are no duplicate documents in Purchasing, but it does not consider finally closed or canceled documents as duplicates. That is, if two documents in Purchasing match an incoming document in the Replace submission, Purchasing replaces the one that is not canceled or finally closed. If one document in Purchasing matches an incoming document, but is canceled or finally closed, Purchasing still replaces it with the new document.

Update

A file with an action code of Update updates the following information on already-submitted blanket purchase agreements or catalog quotations without creating completely new documents:

- Unit Price

  If the price update exceeds the price tolerance you set, the price and price breaks are not updated until or unless the buyer accepts the price update in the Exceeded Price Tolerances window. Any other updates, however, are made to the document when the price/sales catalog is imported.

  The price is updated on the document only, not in the item master. Only the item description is updated in the item master, if you've enabled item description updates as described in Setting Up the Purchasing Documents Open Interface on page 8-39.

- Item Description

- UOM
- Price Breaks (for blanket purchase agreements)
- Expiration Date (for blanket purchase agreements)
- URL descriptive flexfield (if you have one)

  A URL descriptive flexfield provides a supplier URL for additional information about an item.

Expired lines in Purchasing (lines whose Expiration Date has been reached) do not get updated. An Update submission for an expired line is treated as a new line to be added to the blanket purchase agreement. Finally closed or canceled documents, or those that are no longer effective, do not get updated.

Choose Update when you want to update the fields listed above on existing documents, and you want to preserve the existing documents' sourcing rules. For example, an Update submission can be used for daily, weekly, or even quarterly updates to existing documents, while for annual or bi-annual updates, a Replace submission may be better for your business needs.

When the Purchasing Documents Open Interface updates a line on a blanket purchase agreement, it does not update the open release for that line. Only future releases use the updated information.

> **Note:** The Purchasing Documents Open Interface cannot update the UOM on an agreement line for which an open release exists. In this case, the Purchasing Documents Open Interface uses the Expiration Date field to expire the line on the agreement, and creates a new line with the updated UOM, which will be used on future releases.

The Purchasing Documents Open Interface updates documents by doing the following:

- First, it identifies the catalog quotation or blanket purchase agreement that needs to be updated by comparing the supplier document number (VENDOR_DOC_NUM in the PO_HEADERS_INTERFACE table) with the existing supplier document number (VENDOR_ORDER_NUM for a blanket agreement or QUOTE_VENDOR_QUOTE_NUMBER for a catalog quotation in the PO_HEADERS table). The supplier (VENDOR_ID or VENDOR_NAME) and document type (DOCUMENT_TYPE_CODE) must also match. The Purchasing

Documents Open Interface matches only to currently or future-effective documents that are not canceled or finally closed.

- Next, the Purchasing Documents Open Interface processes all the changed lines in the PO_LINES_INTERFACE table. It identifies which lines on the existing documents need to be updated by matching the following, in order:

  – Supplier item number

  – Item number used by your organization, revision number, and item category

  – Item description and item category

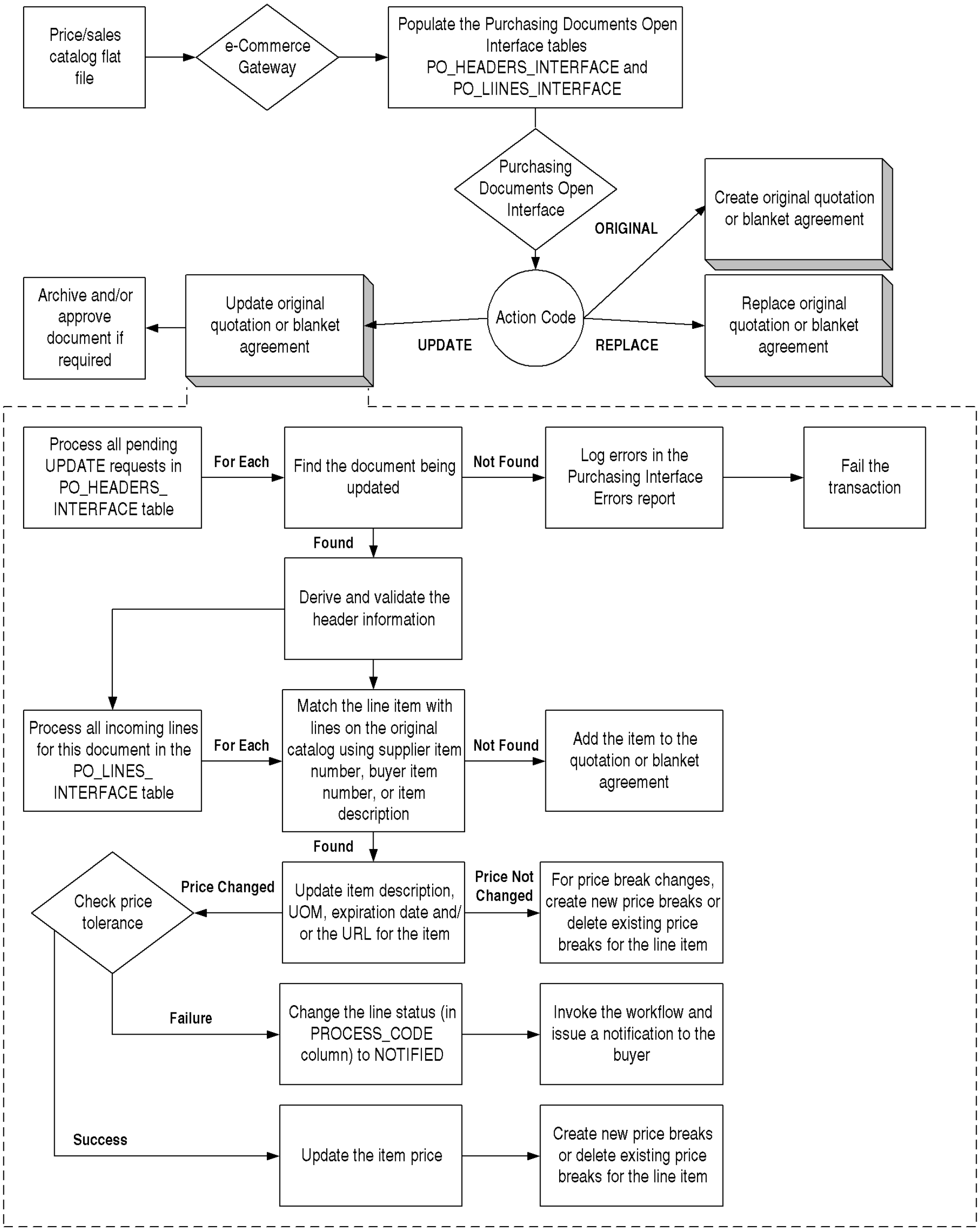
  If it can't match the supplier item number, it tries to match the item number used by your organization (along with the revision number and item category), and so on.

  If more than one line on the existing document matches the incoming line, the Purchasing Documents Open Interface updates the first line on the existing catalog quotation or blanket purchase agreement that matches the incoming line. This first line is also the line that is picked up for sourcing, as long as it has not expired.

  For a one-time item, the item description is updated only if a supplier item number (VENDOR_PRODUCT_NUM) is provided and can be used to find a matching line on the original document.

The following figure shows the flow of an Update price/sales catalog submission to Purchasing:

Figure 8–3 Update Submission Process


Price/sales catalog flat file
e-Commerce Gateway
Populate the Purchasing Documents Open Interface tables PO_HEADERS_INTERFACE and PO_LIINES_INTERFACE
Purchasing Documents Open Interface
ORIGINAL
Create original quotation or blanket agreement
Action Code
UPDATE
REPLACE
Replace original quotation or blanket agreement
Update original quotation or blanket agreement
Archive and/or approve document if required
Process all pending UPDATE requests in PO_HEADERS_ INTERFACE table
For Each
Find the document being updated
Not Found
Log errors in the Purchasing Interface Errors report
Fail the transaction
Found
Derive and validate the header information
Process all incoming lines for this document in the PO_LINES_ INTERFACE table
For Each
Match the line item with lines on the original catalog using supplier item number, buyer item number, or item description
Not Found
Add the item to the quotation or blanket agreement
Found
Check price tolerance
Price Changed
Update item description, UOM, expiration date and/ or the URL for the item
Price Not Changed
For price break changes, create new price breaks or delete existing price breaks for the line item
Failure
Change the line status (in PROCESS_CODE column) to NOTIFIED
Invoke the workflow and issue a notification to the buyer
Success
Update the item price
Create new price breaks or delete existing price breaks for the line item

Sourcing

When you import price/sales catalog information into Purchasing, you have the option of choosing Yes or No in the Create Sourcing Rules field in the Parameters window to enable Purchasing to create sourcing rules out of the supplier, item, and document information that the supplier sends.

If you choose Yes to create sourcing rules in an Original or Replace submission, Purchasing checks if a sourcing rule is assigned to the item at the item level and does the following:

- If no sourcing rules exist for the item, Purchasing generates a sourcing rule automatically, allocating 100 percent to the supplier providing the information.
- If a sourcing rule exists for the item, Purchasing compares the effectivity dates of the incoming document with those of the existing sourcing rule for the item. To ensure that only one sourcing rule is used for the item, Purchasing does the following:
  - If the effectivity dates of the incoming document are the same as the existing sourcing rule's effectivity dates, Purchasing checks if the supplier is in the sourcing rule. If not, Purchasing adds the supplier to the existing sourcing rule with an allocation of 0 percent. Later, you can query the sourcing rule and define your own percentage splits between suppliers.
  - If the effectivity dates of the incoming document are different than the existing sourcing rule's effectivity dates, but are within or overlap the existing effectivity dates, then a new sourcing rule is not created, so as not to conflict with the existing sourcing rule.
  - If the effectivity dates of the incoming document do not overlap the existing sourcing rule's effectivity dates, Purchasing updates the item's sourcing rule with the new effectivity dates, adding the supplier at an allocation of 100 percent.
- Purchasing checks for an Approved Supplier List entry for the item and supplier/site combination. If an entry exists, Purchasing adds the document to the entry. If an entry does not exist, Purchasing creates a new entry with the new source document.

If you choose Yes to create sourcing rules in an Update submission, new sourcing information is created (as described above) only if the Update submission results in a new line being created. See: Adding or Deleting Lines in an Update Submission on page 8-38.

Price Breaks

All action codes (Original, Replace, and Update) support the importing of price breaks through the Purchasing Documents Open Interface.

If one price break is rejected, the whole line to which the price break belongs is rejected.

For Original or Replace Submissions

The following, additional columns are required in the PO_LINES_INTERFACE table if you want to import price break information:

- LINE_NUM
- SHIPMENT_NUM
- QUANTITY
- UNIT_PRICE

If you are importing price break information through catalog quotations, you can also optionally populate the following columns in the PO_LINES_INTERFACE table:

- MIN_ORDER_QUANTITY
- MAX_ORDER_QUANTITY

Recall that the PO_LINES_INTERFACE table contains both line and shipment information, and imports data into both the PO_LINES and PO_LINE_LOCATIONS tables in Purchasing. To create two price breaks corresponding to one blanket agreement or quotation line, you would create two records in the PO_LINES_INTERFACE table. That is, one header-level record in the PO_HEADERS_INTERFACE table would have two records in the PO_LINES_INTERFACE table, and both of these line records would have the same INTERFACE_HEADER_ID:

- One header-level record (row) in the PO_HEADERS_INTERFACE table corresponds to:
  - **Line 1:** one line-level record (row) in the PO_LINES_INTERFACE table, with Shipment 1 information included
  - **Line 1:** the same line-level record (another row) in the PO_LINES_INTERFACE table, with Shipment 2 information included

When two or more lines with the same line and item number are imported into the PO_LINES_INTERFACE table, the Purchasing Document Open Interface treats the subsequent lines as price breaks belonging to the first line.

For the Update Submission

If the supplier updates an item's price, the Purchasing Documents Open Interface deletes the item's price breaks since they are no longer current with the new price. If the supplier sends new price breaks for an existing line, the current price breaks are deleted and the new price breaks sent by the supplier are created.

Just as with an Original or Replace submission, in an Update submission, when two or more lines with the same line and item number are imported into the PO_LINES_INTERFACE table, the Purchasing Document Open Interface treats the subsequent lines as price breaks belonging to the first line.

Attention: In an Update submission, the order of the line and its price breaks is important. If you are writing your own program to populate the interface tables, make sure that the price break lines (lines with the same line and item number, but different shipment numbers) follow the line to which they belong in the interface table. Otherwise, the Purchasing Documents Open Interface will treat them as duplicate lines (and update the previous line with the second). Use the INTERFACE_LINE_ID numbers to order price break (shipment) lines after their corresponding lines.

Adding or Deleting Lines in an Update Submission

The Update submission enables the supplier to add or expire lines on existing blanket purchase agreements or catalog quotations.

The Purchasing Documents Open Interface handles this automatically. If it cannot find a match between the incoming line in the Update submission and an existing line in Purchasing, it adds the line. To expire a line, the supplier must send an updated Expiration Date for the line. You can expire lines on blanket purchase agreements only.

Note: Once a line has expired, the supplier cannot send more updates to it. If the supplier does send an update to an expired line, the update is treated as a new line to be added to the blanket purchase agreement.

Revision Numbering and Archiving

For the Update action code, the document revision number is increased every time the unit Price, item Description, UOM, or Expiration Date is updated, or when a new line is added (unless Archive on Approval is chosen in the Document Types window and *PO: Archive Catalog on Approval* is set to No).

For all action codes, a document is archived upon approval or printing, depending on which option you chose in the Document Types window. When importing price/sales catalog information, you also have the option of choosing Approved or Incomplete in the Approval Status field in the Parameters window. See the table on page 8-44.

The profile option *PO: Archive Catalog on Approval* enables you to choose whether Purchasing archives blanket purchase agreements in a price/sales catalog submission upon approval. (Quotations are not archived in Purchasing.)

If Archive on Approval is chosen in the Document Types window in Purchasing *and* the document is imported as Approved, then:

- Setting *PO: Archive Catalog on Approval* to Yes archives the price/sales catalog documents once they are approved—in this case, as soon as you import them.
- Setting *PO: Archive Catalog on Approval* to No does not archive the price/sales catalog documents. This may be helpful if you receive frequent price/sales catalog submissions and do not want Purchasing to take up extra space archiving every change.

If the document is imported as Incomplete (see the table on page 8-44), it does not matter how the *PO: Archive Catalog on Approval* profile option is set. Purchasing archives the document later upon approval or later upon printing, depending on the option chosen in the Document Types window.

If Archive on Print is chosen in the Document Types window, Purchasing archives the imported price/sales catalog documents once they are printed.

By default the *PO: Archive Catalog on Approval* profile option is set to No. The user can update this profile option. It can also be updated at the system administrator user, responsibility, application, and site levels.

The blanket purchase agreement is archived and its revision number increased only after all of its lines have been either accepted or rejected in the Exceeded Price Tolerances window.

> **Note:** You can monitor changes to blanket purchase agreements by using the PO Change History feature. See: Viewing Purchase Order Changes, *Oracle Purchasing User's Guide*.

See Also

Document Revision Numbering, *Oracle Purchasing User’s Guide*

Setting Up the Purchasing Documents Open Interface

If you want to import supplier price/sales catalog information or responses to RFQs information into the Purchasing Documents Open Interface using the 832/PRICAT or 843/QUOTES transaction, you need to install and set up e-Commerce Gateway for your organization, including defining your supplier as a trading partner, enabling EDI Catalog Inbound/EDI Response to RFQ transactions for that partner, and setting up code conversion categories and values. See: *Oracle e-Commerce Gateway Implementation Manual.*

The concurrent manager(s) that manages all processing also must be set up and running.

Purchasing Setup

Allow Updating of the Item Master When you allow updating of the item master, the Purchasing Documents Open Interface updates the item description not just on blanket purchase agreements or quotations, but in the item master as well. Only the item description is updated in the item master. If the supplier includes a change to the item description in the price/sales catalog, allowing updating of the item master is required for the transaction to go through.

To allow updating of the item master:

1. Navigate to the Purchasing Options window and, in the Control options region, check Allow Item Description Update. See: Defining Control Options, *Oracle Purchasing User's Guide.*

   This allows updating of item descriptions.

2. Navigate to the Personal Profiles window and make sure that *INV: Default Item Status* is set to Active.

   This allows updating of item status codes at the site level.

Set Up Default Category Sets Make sure default category sets are set up appropriately for both Purchasing and Inventory by performing the following steps:

1. Navigate to the Default Category Sets window by choosing Setup > Items > Categories > Default Category Sets in the Purchasing responsibility.

   Make sure that both Purchasing and Inventory are listed in the Functional Area column and each has a default Category Set defined for it.

2. Navigate to the Category Sets window by choosing Setup > Items > Categories > Category Sets in the Purchasing responsibility.

   Make sure that you have a default category set each for both Purchasing and Inventory in the Category Sets window.

   If you've selected the Enforce List of Valid Categories check box in the Category Sets window, make sure that the Default Category also appears in that List of Valid Categories. If not, enter it in the list.

   See: Defining Category Sets, *Oracle Inventory User's Guide*.

Set the profile option *PO: Archive Catalog on Approval* If you typically archive documents on approval, setting this profile option to No means that Purchasing will not archive those changes made through the Purchasing Documents Open Interface. See: Revision Numbering and Archiving on page 8-39.

Set the profile option *PO: Write Server Output to File* Set this profile option when you are debugging the Purchasing Documents Open Interface. When you import a price/sales catalog with a large number of items (about 100 or more), the concurrent manager details log (viewable through the View Log button in the Submit Request window) can overflow and create errors. The profile option *PO: Write Server Output to File* enables you to write these log details to a flat file in the `$APPL_TOP/log` directory to avoid this overflow.

If you set this profile option to Yes, the log details are written to a flat file, which will not overflow. If you set this profile option to No, the log details are written to the concurrent manager log screen as usual, which can cause overflow problems for large catalogs. If you leave this profile option blank, log details are not written at all, which improves performance. By default, the profile option is left blank.

The user can update this profile option. It can also be updated at the system administrator user, responsibility, application, and site levels.

> **Note:** This profile option applies to the Purchasing Documents Open Interface only.

To write log details to a file using Oracle Applications setup:

1. Set *PO: Write Server Output to File* to Yes.
2. After you run the Purchasing Documents Open Interface, look for a system-generated log file in the `$APPL_TOP/log` directory.

To debug using SQL*Plus, do the following before you run the Purchasing Documents Open Interface:

1. Set *PO: Write Server Output to File* to Yes.
2. Make sure the directory for the log file you want to write to in the next step is set in the environment variable `APPLPTMP` and that it is listed in the `UTL_FILE_DIR` parameter in the `init.ora` file.
3. Specify the file name using `FND_FILE.put_names('logfile', 'outfile', 'directory');`

   For example:

   ```
   fnd_file.put_names('mylog.log', 'myout.out', '/sqlcom/out');
   ```

   This will create a file called `mylog.log` in the `/sqlcom/out` directory. It is better to use `/sqlcom/out` because of write-access issues. If you have problems writing to the file, log into the machine where the database is installed.

Set the Price Tolerance for the Update Submission Define your price tolerance for price increases in an Update submission. Optionally use function security to control buyers' access to the Exceeded Price Tolerances window. See: Monitoring Price Increases on page 8-45.

Set the Workflow Timeout for the Update Submission If you import an Update price/sales catalog, decide whether you want to keep the default Timeout of seven days after notifying the buyer of a price update that exceeded the price tolerance. The default Timeout of seven days means if the buyer does not respond after seven days, the buyer will receive the notification again. If want to change this Timeout period, use the Workflow Builder. See: Price / Sales Catalog Notification Workflow, *Oracle Purchasing User's Guide*.

For the Timeout feature to work, the Workflow Background Process must be running. See: To Schedule Background Engines, *Oracle Workflow Guide*.

Importing the Price/Sales Catalog

The import programs window in e-Commerce Gateway initiates both the EDI Catalog Inbound program (or EDI Response to RFQ Inbound program) to import the data from the supplier, and the Purchasing Documents Open Interface program to import the data into Purchasing.

After you submit the information in the Parameters window, make note of the Request ID number by selecting View My Requests from the Help menu so that you can later view the status of your submission.

The Purchasing Documents Open Interface program is also available separately in the Requests window in Purchasing; it can be run only after you've successfully loaded the price / sales catalog data into the PO_HEADERS_INTERFACE and PO_LINES_INTERFACE tables.

Attention: It is important to check the Purchasing Interface Errors report after you import the price/sales catalog. Because the Purchasing Documents Open Interface saves or errors out line by line, it can accept partial documents. Therefore, to see which document lines were not submitted because of errors, you must check the Purchasing Interface Errors report.

Create Sourcing Rules If you choose Yes in the Create Sourcing Rules field, make sure the Approval Status field for the submitted documents is Approved. Sourcing rules can be created only when the Purchasing documents have a status of Approved. See also: Sourcing on page 8-36.

Approval Status The following table shows the effects of the import Approval Status on a document's current status, when you import an Update price/sales catalog.

Table 8–5 Document Status in Update Submission

Current Document Status	Import Status	
	Import as Incomplete	Import as Approved
Incomplete	Document remains Incomplete	Document changes to Approved
Approved	Document remains Approved	Document remains Approved

Note that while a document is Incomplete, you cannot source from it until it is approved. Even in an Update submission, where you are updating only certain lines on a blanket purchase agreement or catalog quotation, if you choose Incomplete, the entire document, including the lines that weren't updated, is considered Incomplete.

Purging the Open Interface Tables after Importing the Catalog

If you want to purge data in the open interface tables after you have imported the data into Purchasing, use the Purge Purchasing Documents Open Interface Processed Data program, available through the Submit Request window in Purchasing. This program removes data that has been accepted or rejected, not data that is still pending.

If you want to purge data from the Purchasing Interface Errors table, set the Purge Data field in the Purchasing Interface Errors report to Yes; the errors will not reappear the next time you run the report.

See Also

Purchasing Documents Open Interface, *Oracle Purchasing User's Guide*

Purchasing Interface Errors Report, *Oracle Purchasing User's Guide*

Purge Purchasing Documents Open Interface Processed Data, *Oracle Purchasing User's Guide*

Inbound Price/Sales Catalog (832/PRICAT), *Oracle e-Commerce Gateway User's Guide*

Inbound Response to Request for Quote (843/QUOTE), *Oracle e-Commerce Gateway User's Guide*

Monitoring Price Increases

You can optionally set a price update tolerance in the Supplier-Item Attributes window, on the blanket purchase agreement, or in the *PO: Price Tolerance (%) for Catalog Updates* profile option.

If a price update in an Update price/sales catalog exceeds your tolerance, the buyer receives a notification for each affected document in the Notifications Summary window. From there, the buyer can access the Exceeded Price Tolerances window and accept or reject the price increase.

The Purchasing Documents Open Interface Update submission makes all changes to documents except price changes that have exceeded your price tolerance. For these lines only, the Purchasing Documents Open Interface waits for the buyer to accept or reject the price increases in the Exceeded Price Tolerances window before completing or rejecting the price change. All other line changes, however, are made.

Purchasing performs the price tolerance check against the price on the current revision of the document. The price tolerance check is performed only on Update price/sales catalog submissions and only on price increases. If the price decreases, Purchasing does not check the decrease against your tolerance or notify you of the decrease.

The Exceeded Price Tolerances window can also be used with function security to control buyers' access to it.

The Purchasing Documents Open Interface uses Oracle Workflow to handle exceeded price tolerance notifications. A default workflow in Purchasing, the PO Catalog Price Tolerance Notifications workflow, automatically sends a notification to the buyer when the price tolerance has been exceeded. The workflow also provides you with function activities that you can modify to enable the workflow to send automatic notifications to your suppliers when you have rejected a price increase. For detailed information about the workflow, see: Price/Sales Catalog Notification Workflow, *Oracle Purchasing User's Guide*.

▶▶ To limit buyers' access to the Exceeded Price Tolerances window:

❏ Exclude the Accept/Reject Exceeded Price Tolerances subfunction from the buyer's responsibility.

Buyers can then only view the Exceeded Price Tolerances window.

See: Overview of Function Security, *Oracle Applications System Administrator's Guide*. See: Forms and Subfunctions, *Oracle Applications System Administrator's Guide*.

◗◗ To set the price update tolerance:

- ❑ Specify a price update tolerance at any of the following levels:
    - The Price Update Tolerance field on the original blanket purchase agreement, in the Terms and Conditions window.
    - The Price Update Tolerance field in the Supplier-Item Attributes window, at the item-supplier level.
    - The Price Update Tolerance field in the Supplier-Item Attributes window, at the commodity-supplier level.
    - The *PO: Price Tolerance (%) for Catalog Updates* profile option if you want the price tolerance to apply to everything, not just to specific documents, or supplier-item or supplier-commodity levels.

For blanket purchase agreements, Purchasing uses the first price tolerance it finds to determine the price tolerance: it looks first at the document, then the item level in the Approved Supplier List, then the category level in the Approved Supplier List, then the profile option. For quotations, since you cannot define a price tolerance at the document level, Purchasing looks first at the item level in the Approved Supplier List, then the category level in the Approved Supplier List, then the profile option. If you set one price update tolerance, it does not default to the other levels. If you set more than one price update tolerance, Purchasing uses the first one it finds in the order described above. If no Price Update Tolerance is set at any of these levels, then Purchasing performs no price tolerance checking.

The Price Update Tolerance fields and profile option apply only to documents in a price/sales catalog submission.

The Price Tolerance field in the Purchasing Options window has nothing to do with these Price Update Tolerance fields.

Example 8–1 Price Tolerance Checking

If you set the Price Update Tolerance field to 20 at the item-supplier level in the Supplier-Item Attributes window (and you haven't set the price tolerance on the agreement), a price increase of more than 20 percent for that item and supplier will send the buyer a notification. If you set the Price Update Tolerance to 20 on the agreement, a price increase of more than 20 percent on that document will issue a notification. If you set the Price Update Tolerance to 0, and you haven't set it to something else at a lower level, then no automatic price updates are allowed. That is, you will receive a notification of every price increase that occurs at that level.

◗◗ To accept or reject price increases:

1. Navigate to the Exceeded Price Tolerances window, accessible under the Purchase Orders menu.

   You can also navigate to the Exceeded Price Tolerances window from the notification. Look for a notification titled *Price tolerance exceeded during BLANKET update* or *Price tolerance exceeded during QUOTATION update*.

2. Choose Accept or Reject.

   For more information, see: Monitoring Price Increases in a Price/Sales Catalog Update, *Oracle Purchasing User's Guide*. Or see the online help for the window.

Purchasing Documents Open Interface Table Descriptions

Values for the columns in the PO_HEADERS_INTERFACE and PO_LINES_INTERFACE tables can come from multiple sources. Your suppliers can send the data, you can enter data yourself through the Parameters windows in the EDI Catalog Inbound program or EDI Response to RFQ Inbound program, and the Purchasing Documents Open Interface program in Purchasing can derive (or default) some of the data into the Purchasing tables. Most of the columns in these tables correspond to columns in the PO_HEADERS and PO_LINES tables in Purchasing.

Most of the columns that end with ID refer to internal identifier columns that uniquely identify a row in a table in Purchasing. The INTERFACE_HEADER_ID column in the PO_HEADERS_INTERFACE table is the primary key (or unique identifier) for this table that other Purchasing tables can reference. Most other ID columns are foreign keys—or identifiers that point—to other tables in Purchasing. For example, VENDOR_SITE_ID and VENDOR_SITE_CODE point to the PO_VENDOR_SITES table.

Some columns described below are not used currently by the Purchasing Documents Open Interface, but are reserved for future functionality.

The table descriptions below are based on what the Purchasing Documents Open Interface itself requires, whether the data is imported through e-Commerce Gateway or a program you write. The following definitions are used:

- *Required:* The Purchasing Documents Open Interface requires these values at a minimum, whether they are imported through a program you write or through e-Commerce Gateway. For example, the Purchasing Documents Open Interface requires a value for the column INTERFACE_HEADER_ID, but e-Commerce Gateway provides a value automatically.

- *Derived and/or Defaulted:* The Purchasing Documents Open Interface can derive or default columns in this category, depending on whether other values are provided. For example, the column AGENT_ID is a Derived and/or Defaulted column if a valid AGENT_NAME is provided.
- *Optional:* You do not have to enter values for columns in this category.
- *Reserved for Future Use:* As of this release, the Purchasing Documents Open Interface does not validate columns in this category before passing them into Purchasing, but has reserved these columns for future enhancements. Do not enter values in these columns.

See Also

Oracle Purchasing Applications Technical Reference Manual, Release 11i

Purchasing Documents Headers Table Description

The following table describes the PO_HEADERS_INTERFACE table.

Table 8–6 Purchasing Documents Open Interface (Headers)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
INTERFACE_HEADER_ID	Number	x			
BATCH_ID	Number			x	
INTERFACE_SOURCE_CODE	Varchar2				x
PROCESS_CODE	Varchar2			x	
ACTION	Varchar2	x			
GROUP_CODE	Varchar2				x
ORG_ID	Number		x	x	
DOCUMENT_TYPE_CODE	Varchar2	x			
DOCUMENT_SUBTYPE	Varchar2			x	
DOCUMENT_NUM	Varchar2		x	x	
PO_HEADER_ID	Number		x	x	
RELEASE_NUM	Number			x	
PO_RELEASE_ID	Number				x
RELEASE_DATE	Date				x

Table 8–6 Purchasing Documents Open Interface (Headers)

PO_HEADERS_INTERFACE **Column Name**	**Type**	**Required**	**Derived and/or Defaulted**	**Optional**	**Reserved for Future Use**
CURRENCY_CODE	Varchar2		x	x	
RATE_TYPE	Varchar2			x	
RATE_TYPE_CODE	Varchar2		x	x	
RATE_DATE	Date		x	x	
RATE	Number		x	x	
AGENT_NAME	Varchar2			x	
AGENT_ID	Number		x	x	
VENDOR_NAME	Varchar2			x	
VENDOR_ID	Number	x	x		
VENDOR_SITE_CODE	Varchar2	x			
VENDOR_SITE_ID	Number	x	x		
VENDOR_CONTACT	Varchar2			x	
VENDOR_CONTACT_ID	Number		x	x	
SHIP_TO_LOCATION	Varchar2			x	
SHIP_TO_LOCATION_ID	Number		x	x	
BILL_TO_LOCATION	Varchar2			x	
BILL_TO_LOCATION_ID	Number		x	x	
PAYMENT_TERMS	Varchar2			x	
TERMS_ID	Number		x	x	
FREIGHT_CARRIER	Varchar2		x	x	
FOB	Varchar2		x	x	
FREIGHT_TERMS	Varchar2		x	x	
APPROVAL_STATUS	Varchar2			x	
APPROVED_DATE	Date		x		
REVISED_DATE	Date				x
REVISION_NUM	Number			x	
NOTE_TO_VENDOR	Varchar2				x
NOTE_TO_RECEIVER	Varchar2				x

Table 8–6 Purchasing Documents Open Interface (Headers)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
CONFIRMING_ORDER_FLAG	Varchar2			x	
COMMENTS	Varchar2			x	
ACCEPTANCE_REQUIRED_ FLAG	Varchar2		x	x	
ACCEPTANCE_DUE_DATE	Date			x	
AMOUNT_AGREED	Number			x	
AMOUNT_LIMIT	Number				x
MIN_RELEASE_AMOUNT	Number		x	x	
EFFECTIVE_DATE	Date	*conditionally*			
EXPIRATION_DATE	Date	*conditionally*			
PRINT_COUNT	Number		x	x	
PRINTED_DATE	Date				x
FIRM_FLAG	Varchar2				x
FROZEN_FLAG	Varchar2		x	x	
CLOSED_CODE	Varchar2		x	x	
CLOSED_DATE	Date				x
REPLY_DATE	Date		x	x	
REPLY_METHOD	Varchar2				x
RFQ_CLOSE_DATE	Date				x
QUOTE_WARNING_DELAY	Number		x	x	
VENDOR_DOC_NUM	Varchar2	x			
APPROVAL_REQUIRED_FLAG	Varchar2		x	x	
VENDOR_LIST	Varchar2				x
VENDOR_LIST_HEADER_ID	Number				x
FROM_HEADER_ID	Number		x	x	
FROM_TYPE_LOOKUP_CODE	Varchar2		x	x	
USSGL_TRANSACTION_CODE	Varchar2			x	
ATTRIBUTE_CATEGORY	Varchar2			x	

Table 8–6 Purchasing Documents Open Interface (Headers)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
ATTRIBUTE1	Varchar2			x	
ATTRIBUTE2	Varchar2			x	
ATTRIBUTE3	Varchar2			x	
ATTRIBUTE4	Varchar2			x	
ATTRIBUTE5	Varchar2			x	
ATTRIBUTE6	Varchar2			x	
ATTRIBUTE7	Varchar2			x	
ATTRIBUTE8	Varchar2			x	
ATTRIBUTE9	Varchar2			x	
ATTRIBUTE10	Varchar2			x	
ATTRIBUTE11	Varchar2			x	
ATTRIBUTE12	Varchar2			x	
ATTRIBUTE13	Varchar2			x	
ATTRIBUTE14	Varchar2			x	
ATTRIBUTE15	Varchar2			x	
CREATION_DATE	Date		x	x	
CREATED_BY	Number		x	x	
LAST_UPDATE_DATE	Date		x	x	
LAST_UPDATED_BY	Number		x	x	
LAST_UPDATE_LOGIN	Number		x	x	
REQUEST_ID	Number		x	x	
PROGRAM_APPLICATION_ID	Number		x	x	
PROGRAM_ID	Number		x	x	
PROGRAM_UPDATE_DATE	Date		x	x	
REFERENCE_NUM	Varchar2			x	
LOAD_SOURCING_RULES_ FLAG	Varchar2			x	
VENDOR_NUM	Varchar2			x	

Table 8–6 Purchasing Documents Open Interface (Headers)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
FROM_RFQ_NUM	Varchar2			x	
WF_GROUP_ID	Number			x	
PCARD_ID	Number				x
PAY_ON_CODE	Varchar2				x

Following is a description of all of the required and conditionally required columns in the PO_HEADERS_INTERFACE table, and some other columns. Remaining column descriptions can be found in the *Oracle Purchasing Applications Technical Reference Manual, Release 11i.*

INTERFACE_HEADER_ID Required

This column indicates an identifier for the purchase order or catalog header. If you import price/sales catalog information through e-Commerce Gateway, this identifier is provided automatically.

BATCH_ID Optional

When you import the price/sales catalog information through e-Commerce Gateway, it provides a concurrent program grouping identifier for the submission.

INTERFACE_SOURCE_CODE Reserved for Future Use

This column identifies the source (for example, e-Commerce Gateway) of the price/sales catalog data.

PROCESS_CODE Optional

This column indicates the status of a row in the interface table. It accepts values of PENDING, ACCEPTED, REJECTED, or NOTIFIED. A PENDING transaction has not yet been processed. An ACCEPTED transaction has been successfully processed. A REJECTED transaction contains an error which shows up in the Purchasing Interface Errors Report. A NOTIFIED transaction, which includes a price update that exceeded the price tolerance, has been communicated to the buyer through the Notifications Summary window.

When you import price/sales catalog information through e-Commerce Gateway, e-Commerce Gateway defaults a value of PENDING in this column automatically.

Then, the Purchasing Documents Open Interface sets the value to ACCEPTED, REJECTED, or NOTIFIED.

ACTION Required

This column indicates whether the price/sales catalog information is an original (new), replacement, or update file. This column accepts values of ORIGINAL, REPLACE, or UPDATE.

GROUP_CODE Reserved for Future Use

This column indicates an identifier for the batch being imported.

DOCUMENT_TYPE_CODE Required

This column accepts values of BLANKET or QUOTATION. It is required to match incoming documents with existing documents.

VENDOR_NAME or VENDOR_ID Required

VENDOR_NAME indicates the supplier for the document. VENDOR_ID indicates the supplier identifier number. Make sure the supplier is also set up as a trading partner in the e-Commerce Gateway application, if you're importing data through e-Commerce Gateway. If you provide a value for one of these columns, you do not have to provide a value for the other.

VENDOR_SITE_CODE or VENDOR_SITE_ID Required

This column indicates the supplier site for the document. If the supplier has more than one site, the Purchasing Documents Open Interface cannot default a site. The site also needs to be set up in e-Commerce Gateway, if you're importing data through e-Commerce Gateway. If you provide a value for one of these columns, you do not have to provide a value for the other.

EFFECTIVE_DATE Conditionally Required

This column must be populated when replacing an existing purchasing document. The value in this column is used to locate the old price/sales catalog and expire it.

EXPIRATION_DATE Conditionally Required

This column must be populated when replacing an existing purchasing document. The value in this column is used to locate the old price/sales catalog and expire it.

VENDOR_DOC_NUM Required

This column must be populated when replacing or updating an existing purchasing document. It is also required to make sure that documents in an Original submission don't already exist in Purchasing.

LOAD_SOURCING_RULES_FLAG Optional

This column indicates whether to create sourcing rules with the purchasing document. You choose this option in the Parameters window when importing the price/sales catalog.

Purchasing Documents Lines Table Description

The following table describes the PO_LINES_INTERFACE table.

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
INTERFACE_LINE_ID	Number	x			
INTERFACE_HEADER_ID	Number	x			
ACTION	Varchar2				x
GROUP_CODE	Varchar2				x
LINE_NUM	Number		x	x	
PO_LINE_ID	Number		x	x	
SHIPMENT_NUM	Number		x	x	
LINE_LOCATION_ID	Number		x	x	
SHIPMENT_TYPE	Varchar2		x	x	
REQUISITION_LINE_ID	Number				x
DOCUMENT_NUM	Number				x
RELEASE_NUM	Number				x
PO_HEADER_ID	Number		x	x	
PO_RELEASE_ID	Number				x
EXPIRATION_DATE	Date			x	
SOURCE_SHIPMENT_ID	Number				x
CONTRACT_NUM	Varchar2				x

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
LINE_TYPE	Varchar2			x	
LINE_TYPE_ID	Number		x	x	
ITEM	Varchar2	*conditionally*			
ITEM_ID	Number		x	x	
ITEM_REVISION	Varchar2			x	
CATEGORY	Varchar2			x	
CATEGORY_ID	Number		x	x	
ITEM_DESCRIPTION	Varchar2	*conditionally*	x		
VENDOR_PRODUCT_NUM	Varchar2		x	x	
UOM_CODE	Varchar2		x	x	
UNIT_OF_MEASURE	Varchar2		x	x	
QUANTITY	Number		x	x	
COMMITTED_AMOUNT	Number			x	
MIN_ORDER_QUANTITY	Number			x	
MAX_ORDER_QUANTITY	Number			x	
UNIT_PRICE	Number		x	x	
LIST_PRICE_PER_UNIT	Number		x	x	
MARKET_PRICE	Number		x	x	
ALLOW_PRICE_OVERRIDE_ FLAG	Varchar2		x	x	
NOT_TO_EXCEED_PRICE	Number			x	
NEGOTIATED_BY_PREPARER_ FLAG	Varchar2		x	x	
UN_NUMBER	Varchar2			x	
UN_NUMBER_ID	Number		x	x	
HAZARD_CLASS	Varchar2			x	
HAZARD_CLASS_ID	Number		x	x	
NOTE_TO_VENDOR	Varchar2				x

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
TRANSACTION_REASON_CODE	Varchar2				x
TAXABLE_FLAG	Varchar2		x	x	
TAX_NAME	Varchar2		x	x	
TYPE_1099	Varchar2		x		
CAPITAL_EXPENSE_FLAG	Varchar2		x	x	
INSPECTION_REQUIRED_FLAG	Varchar2		x	x	
RECEIPT_REQUIRED_FLAG	Varchar2		x	x	
PAYMENT_TERMS	Varchar2			x	
TERMS_ID	Number		x	x	
PRICE_TYPE	Varchar2		x	x	
MIN_RELEASE_AMOUNT	Number		x	x	
PRICE_BREAK_LOOKUP_CODE	Varchar2		x	x	
USSGL_TRANSACTION_CODE	Varchar2			x	
CLOSED_CODE	Varchar2		x	x	
CLOSED_REASON	Varchar2				x
CLOSED_DATE	Date				x
CLOSED_BY	Number				x
INVOICE_CLOSE_TOLERANCE	Number				x
RECEIVE_CLOSE_TOLERANCE	Number				x
FIRM_FLAG	Varchar2				x
DAYS_EARLY_RECEIPT_ALLOWED	Number			x	
DAYS_LATE_RECEIPT_ALLOWED	Number			x	
ENFORCE_SHIP_TO_LOCATION_CODE	Varchar2			x	
ALLOW_SUBSTITUTE_RECEIPTS_FLAG	Varchar2			x	
RECEIVING_ROUTING	Varchar2			x	

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE **Column Name**	**Type**	**Required**	**Derived and/or Defaulted**	**Optional**	**Reserved for Future Use**
RECEIVING_ROUTING_ID	Number			x	
QTY_RCV_TOLERANCE	Number		x	x	
OVER_TOLERANCE_ERROR_FLAG	Varchar2			x	
QTY_RCV_EXCEPTION_CODE	Varchar2		x	x	
RECEIPT_DAYS_EXCEPTION_CODE	Varchar2			x	
SHIP_TO_ORGANIZATION_CODE	Varchar2			x	
SHIP_TO_ORGANIZATION_ID	Number		x	x	
SHIP_TO_LOCATION	Varchar2			x	
SHIP_TO_LOCATION_ID	Number		x	x	
NEED_BY_DATE	Date			x	
PROMISED_DATE	Date			x	
ACCRUE_ON_RECEIPT_FLAG	Varchar2				x
LEAD_TIME	Number			x	
LEAD_TIME_UNIT	Varchar2			x	
PRICE_DISCOUNT	Number			x	
FREIGHT_CARRIER	Varchar2		x	x	
FOB	Varchar2		x	x	
FREIGHT_TERMS	Varchar2		x	x	
EFFECTIVE_DATE	Date	*conditionally*			
EXPIRATION_DATE	Date	*conditionally*			
FROM_HEADER_ID	Number			x	
FROM_LINE_ID	Number			x	
FROM_LINE_LOCATION_ID	Number			x	
LINE_ATTRIBUTE_CATEGORY_LINES	Varchar2			x	
LINE_ATTRIBUTE1	Varchar2			x	
LINE_ATTRIBUTE2	Varchar2			x	

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
LINE_ATTRIBUTE3	Varchar2			x	
LINE_ATTRIBUTE4	Varchar2			x	
LINE_ATTRIBUTE5	Varchar2			x	
LINE_ATTRIBUTE6	Varchar2			x	
LINE_ATTRIBUTE7	Varchar2			x	
LINE_ATTRIBUTE8	Varchar2			x	
LINE_ATTRIBUTE9	Varchar2			x	
LINE_ATTRIBUTE10	Varchar2			x	
LINE_ATTRIBUTE11	Varchar2			x	
LINE_ATTRIBUTE12	Varchar2			x	
LINE_ATTRIBUTE13	Varchar2			x	
LINE_ATTRIBUTE14	Varchar2			x	
LINE_ATTRIBUTE15	Varchar2			x	
SHIPMENT_ATTRIBUTE_ CATEGORY	Varchar2			x	
SHIPMENT_ATTRIBUTE1	Varchar2			x	
SHIPMENT_ATTRIBUTE2	Varchar2			x	
SHIPMENT_ATTRIBUTE3	Varchar2			x	
SHIPMENT_ATTRIBUTE4	Varchar2			x	
SHIPMENT_ATTRIBUTE5	Varchar2			x	
SHIPMENT_ATTRIBUTE6	Varchar2			x	
SHIPMENT_ATTRIBUTE7	Varchar2			x	
SHIPMENT_ATTRIBUTE8	Varchar2			x	
SHIPMENT_ATTRIBUTE9	Varchar2			x	
SHIPMENT_ATTRIBUTE10	Varchar2			x	
SHIPMENT_ATTRIBUTE11	Varchar2			x	
SHIPMENT_ATTRIBUTE12	Varchar2			x	
SHIPMENT_ATTRIBUTE13	Varchar2			x	

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
SHIPMENT_ATTRIBUTE14	Varchar2			x	
SHIPMENT_ATTRIBUTE15	Varchar2			x	
LAST_UPDATE_DATE	Date			x	
LAST_UPDATED_BY	Number		x	x	
LAST_UPDATE_LOGIN	Number		x	x	
CREATION_DATE	Date		x	x	
CREATED_BY	Number		x	x	
REQUEST_ID	Number		x	x	
PROGRAM_APPLICATION_ID	Number		x	x	
PROGRAM_ID	Number		x	x	
PROGRAM_UPDATE_DATE	Date		x	x	
ORGANIZATION_ID	Number		x	x	
ITEM_ATTRIBUTE_CATEGORY	Varchar2			x	
ITEM_ATTRIBUTE1	Varchar2			x	
ITEM_ATTRIBUTE2	Varchar2			x	
ITEM_ATTRIBUTE3	Varchar2			x	
ITEM_ATTRIBUTE4	Varchar2			x	
ITEM_ATTRIBUTE5	Varchar2			x	
ITEM_ATTRIBUTE6	Varchar2			x	
ITEM_ATTRIBUTE7	Varchar2			x	
ITEM_ATTRIBUTE8	Varchar2			x	
ITEM_ATTRIBUTE9	Varchar2			x	
ITEM_ATTRIBUTE10	Varchar2			x	
ITEM_ATTRIBUTE11	Varchar2			x	
ITEM_ATTRIBUTE12	Varchar2			x	
ITEM_ATTRIBUTE13	Varchar2			x	
ITEM_ATTRIBUTE14	Varchar2			x	
ITEM_ATTRIBUTE15	Varchar2			x	

Table 8–7 Purchasing Documents Open Interface (Lines)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
UNIT_WEIGHT	Number			x	
WEIGHT_UOM_CODE	Varchar2			x	
VOLUME_UOM_CODE	Varchar2			x	
UNIT_VOLUME	Number			x	
TEMPLATE_ID	Number		x	x	
TEMPLATE_NAME	Varchar2			x	
LINE_REFERENCE_NUM	Varchar2			x	
SOURCING_RULE_NAME	Varchar2			x	
TAX_STATUS_INDICATOR	Varchar2				x
PROCESS_CODE	Varchar2	*for internal use only*			
PRICE_CHG_ACCEPT_FLAG	Varchar2	*for internal use only*			
PRICE_BREAK_FLAG	Varchar2	*for internal use only*			
PRICE_UDATE_TOLERANCE	Number			x	
TAX_USER_OVERRIDE_FLAG	Varchar2			x	
TAX_ID	Number			x	
TAX_CODE_ID	Number			x	

Following is a description of all of the required and conditionally required columns in the PO_LINES_INTERFACE table, and some other columns. Remaining column descriptions can be found in the *Oracle Purchasing Applications Technical Reference Manual, Release 11i.*

INTERFACE_LINE_ID Required

This column indicates the unique identifier of the line record in the PO_LINES_INTERFACE table. If you import price/sales catalog information through e-Commerce Gateway, this identifier is provided automatically.

INTERFACE_HEADER_ID Required

This column indicates the unique identifier of the header record to which this line belongs. If you import price/sales catalog information through e-Commerce Gateway, this identifier is provided automatically.

GROUP_CODE Reserved for Future Use

This column indicates an identifier for the batch being imported.

EFFECTIVE_DATE Conditionally Required

If you choose to create sourcing rules with the imported price/sales catalog information, then you need to provide a value in this column.

EXPIRATION_DATE Conditionally Required

In an Original or Replace submission: If you choose to create sourcing rules with the imported price/sales catalog information, then you need to provide a value in this column.

In an Update submission: When updating an existing line, the value in this column is used to expire the line. When adding a new line, the value in this column is required if you choose to create sourcing rules with the imported price/sales catalog information.

ITEM or ITEM_DESCRIPTION Conditionally Required

If you want to create items in the item master, then you need to supply the item information.

Both an ITEM and an ITEM_DESCRIPTION are required if an item is being created in the item master. If an item is being updated, the ITEM_DESCRIPTION is required.

SOURCING_RULE_NAME Optional

If sourcing rules are being used, this column indicates the name of the sourcing rule.

PROCESS_CODE Internal Use Only

This column indicates the status of a row in the interface table. It is updated internally with values of null (which means PENDING), ACCEPTED, or NOTIFIED. A PENDING transaction has not yet been processed. An ACCEPTED transaction has been successfully processed. A NOTIFIED transaction, which includes a price update that exceeded the price tolerance, has been communicated to the buyer through the Notifications Summary window. The Purchasing Documents Open Interface sets the value of this column to ACCEPTED or NOTIFIED.

This column is for internal use only. Unless you are debugging the open interface, do not use a program or script to update this column yourself.

PRICE_CHG_ACCEPT_FLAG Internal Use Only

This column indicates internally whether a price has changed in an Update submission. If the PROCESS_CODE value is NOTIFIED, NULL means that the item price update is pending—waiting for the buyer to respond with Accept or Reject in the Exceeded Price Tolerances window. Y means that the price update was accepted. N means that it was rejected.

This column is for internal use only. Unless you are debugging the open interface, do not use a program or script to update this column yourself.

PRICE_BREAK_FLAG Internal Use Only

This column indicates internally whether a record in an Update submission includes price breaks.

This column is for internal use only. Unless you are debugging the open interface, do not use a program or script to update this column yourself.

Derivation

In general, the same derivation and defaulting rules apply to the interface tables as apply when you enter information in the Purchase Orders or Quotations windows. For example, the column ITEM_DESCRIPTION is derived or defaulted only if a valid ITEM or ITEM_ID is provided.

The Purchasing Documents Open Interface supports column value passing by user value; for example, if you provide a VENDOR_NAME or VENDOR_NUM, the VENDOR_ID is derived. Purchasing uses the derivation source according to the following rules:

- Key (ID) columns always override value columns. If you populate both the key column and the corresponding value column, then the key column is always used for processing. For example, if VENDOR_NAME and VENDOR_ID contradict each other, VENDOR_ID is used.
- Derivation is performed before defaulting, and generally overrides defaulting. (Derivation refers to deriving a full value from a partial value given; defaulting refers to using a default value in Purchasing when no value is given.) For example, if you load the SHIP_TO_LOCATION value in the interface tables, Purchasing derives the SHIP_TO_LOCATION_ID from it instead of from the default ship-to information associated with your supplier.

Defaulting

The Purchasing Documents Open Interface supports the same defaulting mechanisms as the Purchasing document entry windows. Defaults can come from many sources, such as the Purchasing Options, Financial Options, Suppliers, and Master Item (or Organization Items) windows.

Defaulting rules are applied as follows:

- Defaults do not override values that you specify.
- Default values that are no longer active or valid are not used.

Validation

The Purchasing Documents Open Interface does not validate those columns described as "Reserved for Future Use" on the previous pages.

Standard Validation

Purchasing validates all required columns in the interface table. For specific information on the data implied by these columns, see your *Oracle Purchasing Applications Technical Reference Manual, Release 11i,* for details.

Other Validation

The Purchasing Documents Open Interface performs the same validation as the Purchasing document entry windows before allowing the data to be committed to the base tables.

If multiple errors are detected, each error is written to the PO_INTERFACE_ ERRORS table and displayed in the Purchasing Interface Errors Report.

Not only are all required columns validated so that they are populated with acceptable values, but also errors are signaled for those columns that have values but should not. For example, if a RATE_TYPE does not need to be defined (because exchange rate information is not needed) but contains a value, an error will be signaled.

Resolving Failed Purchasing Interface Rows

Error Messages

Purchasing may display specific error messages during interface processing. For more details on these messages, please see the *Oracle Applications Message Reference Manual*, in HTML format on the documentation CD-ROM for Release 11*i*.

A Note about Debugging

If you receive an error in a document, you can fix the error and resubmit the document again by reusing a header record in the PO_HEADERS_INTERFACE table using SQL*Plus, if you’re using a test environment. If you do this, set the PROCESS_CODE column in the PO_HEADERS_INTERFACE table to PENDING and be sure to reset the following columns to NULL for all lines belonging to that header in the PO_LINES_INTERFACE table:

- PROCESS_CODE
- PRICE_CHG_ACCEPT_FLAG
- PRICE_BREAK_FLAG

Viewing Failed Transactions

You can report on all rows that failed validation by using the Purchasing Interface Errors Report. For each row in the Purchasing Documents Open Interface tables that fails validation, the Purchasing Documents Open Interface creates one or more rows with error information in the PO_INTERFACE_ERRORS table. The Purchasing Interface Errors Report lists all the columns in the PO_INTERFACE_ERRORS table that failed validation along with the reason for the failure. This report is generated through the Submit Request window and processed as other standard reports in Purchasing.

The following table shows the error messages and their meaning:

Table 8–8 Purchasing Documents Open Interface Error Messages

Error Message	Meaning
PO_PDOI_AMT_LIMIT_LT_AGREED	Amount Limit (VALUE= &AMOUNT_LIMIT) is less than Amount Agreed (VALUE=&VALUE).
PO_PDOI_AMT_LIMIT_LT_RELEASE	Amount Limit (VALUE= &AMOUNT_LIMIT) is less than Minimum Release Amount (VALUE=&VALUE).
PO_PDOI_AMT_LIMIT_LT_TOTREL	Amount Limit (VALUE= &AMOUNT_LIMIT) is less than Total Amount Released (VALUE=&VALUE).

Table 8–8 Purchasing Documents Open Interface Error Messages

Error Message	Meaning
PO_PDOI_CATG_ALREADY_EXISTS	Catalog being created with supplier document number [DOC_NUMBER] already exists.
PO_PDOI_COLUMN_NOT_NULL	Column &COLUMN_NAME should not be NULL.
PO_PDOI_COLUMN_NOT_ZERO	Column &COLUMN_NAME should be 0.
PO_PDOI_COLUMN_NULL	Column &COLUMN_NAME (VALUE=&VALUE) must be NULL.
PO_PDOI_DERV_ERROR	Derivation Error: &COLUMN_NAME (VALUE= &VALUE) specified is invalid.
PO_PDOI_DERV_PART_NUM_ERROR	Cannot derive ITEM_ID for the specified buyer item_ number or VENDOR_PRODUCT_NUM.
PO_PDOI_DIFF_ITEM_DESC	Pre-defined item description cannot be changed for this item.
PO_PDOI_DOC_NUM_UNIQUE	Document Num must have a unique value. &VALUE already exists.
PO_PDOI_EFF_DATE_GT_HEADER	Effective Date (VALUE =&VALUE) specified should not be less than the effective date specified.
PO_PDOI_EXCEED_PRICE_NULL NOT TO EXCEED PRICE (VALUE= &VALUE)	Must be NULL if ALLOW_PRICE_OVERRIDE_FLAG is N.
PO_PDOI_INVALID_ACTION	Action (VALUE= &VALUE) is invalid.
PO_PDOI_INVALID_BILL_LOC_ID	Bill-To Location ID (VALUE=&VALUE) is not valid.
PO_PDOI_INVALID_BUYER	Buyer (VALUE=&VALUE) specified is not a valid buyer.
PO_PDOI_INVALID_CATEGORY_ID	CATEGORY ID (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_CURRENCY	Currency Code (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_DISCOUNT	DISCOUNT (VALUE =&VALUE) specified is invalid.
PO_PDOI_INVALID_DOC_NUM	Document Number (VALUE= &VALUE) specified is invalid.
PO_PDOI_INVALID_DOC_STATUS	Sourcing rule can be created only if document is loaded as an approved document.
PO_PDOI_INVALID_FLAG_VALUE	&COLUMN_NAME (VALUE =&VALUE) is invalid. It can either be Y or N.
PO_PDOI_INVALID_FOB	FOB (VALUE=&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_FREIGHT_CARR	FREIGHT CARRIER (VALUE =&VALUE) specified is inactive or invalid.

Table 8–8 Purchasing Documents Open Interface Error Messages

Error Message	Meaning
PO_PDOI_INVALID_FREIGHT_TERMS	FREIGHT TERMS (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_HAZ_ID	HAZARD CLASS ID (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_INTER_LINE_REC	Record specified in PO_LINES_INTERFACE is invalid. It is neither a new record in PO_LINES nor PO_LINE_LOCATIONS.
PO_PDOI_INVALID_ITEM_FLAG	Item Flag (VALUE= &VALUE) is invalid.
PO_PDOI_INVALID_ITEM_ID	ITEM ID (VALUE =&VALUE) is not a valid purchasable item.
PO_PDOI_INVALID_ITEM_REVISION	REVISION NUM (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_ITEM_UOM_CODE	ITEM UOM CODE (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_LEAD_TIME	Lead Time Unit (VALUE=&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_LINE_TYPE_ID	LINE TYPE ID (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_LINE_TYPE_INFO	&COLUMN_NAME (VALUE=&VALUE) must match the value from the PO_LINE_TYPES table (VALUE=&LINE_TYPE).
PO_PDOI_INVALID_LOCATION_REC	Information specified in PO_LINES_INTERFACE table does not match the parent record in PO_LINES table.
PO_PDOI_INVALID_NUM_OF_LINES	&COLUMN_NAME There should be at least one line per document.
PO_PDOI_INVALID_OP_ITEM_ID	ITEM ID (VALUE =&VALUE) is not a valid purchasable and outside operational item.
PO_PDOI_INVALID_ORIG_CATALOG	&DOC_NUMBER specified is not a valid original catalog.
PO_PDOI_INVALID_PAY_TERMS	PAYMENT TERMS (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_PRICE	NOT TO EXCEED PRICE. (VALUE= &VALUE) has to be greater or equal to UNIT PRICE (VALUE=&UNIT_PRICE).
PO_PDOI_INVALID_PRICE_BREAK	PRICE BREAK LOOKUP CODE (VALUE=&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_PRICE_TYPE	PRICE TYPE (VALUE =&VALUE) specified is inactive or invalid.

Table 8–8 Purchasing Documents Open Interface Error Messages

Error Message	Meaning
PO_PDOI_INVALID_QUOTE_TYPE_CD	Document Subtype (VALUE= &VALUE) specified is invalid.
PO_PDOI_INVALID_RATE	The rate value (VALUE=&VALUE) specified is invalid.
PO_PDOI_INVALID_RATE_TYPE	Rate Type (VALUE =&VALUE) specified is invalid.
PO_PDOI_INVALID_RCV_EXCEP_CD	RCV EXCEPTION CODE (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_REPLY_METHOD	REPLY METHOD (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_SHIPMENT_TYPE	SHIPMENT TYPE (VALUE= &TYPE) specified is not valid for TYPE LOOKUP CODE (VALUE=&VALUE)
PO_PDOI_INVALID_SHIP_LOC_ID	SHIP_TO_LOCATION_ID (VALUE=&VALUE) is not valid.
PO_PDOI_INVALID_SHIP_TO_LOC_ID	SHIP_TO_LOCATION_ID (VALUE=&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_SHIP_TO_ORG_ID	SHIP_TO_ORGANIZATION_ID (VALUE=&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_START_DATE	Effective Date (VALUE =&VALUE) specified should be less than the end date specified.
PO_PDOI_INVALID_STATUS	Approval Status specified is invalid.
PO_PDOI_INVALID_TAX_NAME	TAX NAME (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_TEMPLATE_ID	TEMPLATE ID (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_TYPE_LKUP_CD	Document Type Code (VALUE =&VALUE) specified is invalid.
PO_PDOI_INVALID_UN_NUMBER_ID	UN NUMBER ID (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_UOM_CODE	UNIT OF MEASURE (VALUE =&VALUE) specified is inactive or invalid.
PO_PDOI_INVALID_USSGL_TXN_CODE	USSGL Transaction Code (VALUE =&VALUE) specified is invalid.
PO_PDOI_INVALID_VALUE	&COLUMN_NAME must have a value of &VALUE.
PO_PDOI_INVALID_VDR_CNTCT	Supplier Contact (VALUE=&VALUE) is not an active and valid contact for the specified supplier site.
PO_PDOI_INVALID_VENDOR	Supplier (VALUE=&VALUE) specified is invalid or inactive.

Table 8–8 Purchasing Documents Open Interface Error Messages

Error Message	Meaning
PO_PDOI_INVALID_VENDOR_SITE	Supplier Site (VALUE=&VALUE) is not an active and valid purchasing supplier site.
PO_PDOI_INVAL_MULT_ORIG_CATG	Multiple catalogs can be found with the same document number (&DOC_NUMBER).
PO_PDOI_ITEM_NOT_NULL	ITEM_ID should not be null for outside operation LINE_TYPE.
PO_PDOI_ITEM_RELATED_INFO	&COLUMN_NAME (VALUE=&VALUE) specified is inactive or invalid for ITEM_ID (VALUE=&ITEM).
PO_PDOI_ITEM_UPDATE_NOT_ALLOW	Item attribute(s) required update. However, this execution does not allow item update/creation.
PO_PDOI_LINE_ID_UNIQUE	LINE_ID must have a unique value. &VALUE already exists.
PO_PDOI_LINE_LOC_ID_UNIQUE	LINE_LOCATION_ID must be unique. &VALUE already exists.
PO_PDOI_LINE_NUM_UNIQUE	LINE_NUM must have a unique value. &VALUE already exists.
PO_PDOI_LT_ZERO	&COLUMN_NAME (VALUE =&VALUE) specified is less than zero.
PO_PDOI_MULT_BUYER_PART	Multiple buyer parts are found which match the specified Item Num (VALUE=&VALUE).
PO_PDOI_NO_DATA_FOUND	No rate found for currency_code (VALUE=&CURRENCY) and RATE_TYPE_CODE (VALUE=&RATE_TYPE).
PO_PDOI_OVERLAP_AUTO_RULE	Sourcing rule (VALUE=&START_DATE) and (VALUE=&END_DATE) overlaps with an existing sourcing rule.
PO_PDOI_PO_HDR_ID_UNIQUE	PO_HEADER_ID must be unique. (VALUE = &VALUE) already exists.
PO_PDOI_PRICE_BRK_AMT_BASED_LN	Cannot create price breaks for amount-based lines in a BLANKET order agreement.
PO_PDOI_QT_MIN_GT_MAX	Minimum Quantity (VALUE =&MIN) specified is greater than maximum Quantity (VALUE =&MAX).
PO_PDOI_RATE_INFO_NULL	The RATE TYPE, RATE_DATE and RATE must be null because the purchasing document’s currency is the same as your base (functional) currency. Therefore, exchange rate information is not needed.
PO_PDOI_RULE_NAME_UNIQ	Rule Name (VALUE= &VALUE) and Item Id (ID =&VALUE) should be unique.
PO_PDOI_SHIPMENT_NUM_UNIQUE	Shipment Num must have a unique value. &VALUE already exists.

Table 8–8 Purchasing Documents Open Interface Error Messages

Error Message	Meaning
PO_PDOI_SPECIF_DIFF_IN_LINES	&COLUMN_NAME (VALUE= &PO_HEADER_ID) specified in line is different from (VALUE=&VALUE) in header.
PO_PDOI_VALUE_NUMERIC	&COLUMN_NAME (VALUE= &VALUE) needs to be a numeric value.
ORIGINAL_RFQ_NUM is invalid	The original RFQ number that is transmitted already exists. Select another RFQ number.
Category ID is invalid for Item ID.	The category ID transmitted with the Item ID does not match the category ID already set up in the system for that item. If it is null, make sure that Purchasing is defaulting a category ID and that the category ID is enabled. Default item categories are specified in Setup > Items > Category > Category Set. Make sure that the default item category is one of the values in the rows.

Fixing Failed Transactions

Some examples of errors could be that the supplier's information does not conform with Purchasing data requirements (for example, date fields are in an incorrect format), cross-reference rules set up between you and your supplier are inaccurate, or your own Purchasing or Oracle Applications data is not up to date. If errors exist in the supplier's data, ask the supplier to correct and resend the data.

Other errors could be the result of the following:

- If you create sourcing rules along with the imported data, make sure the documents in the data are submitted as approved. Sourcing rules can be created only when the Purchasing documents have a status of Approved.
- Flexfields may need to be frozen and recompiled. Navigate to the Descriptive Flexfield Segments window by choosing Setup > Flexfields > Descriptive > Segments. See: Defining Descriptive Flexfield Structures, *Oracle Applications Flexfields Guide*.

See Also

Purchasing Interface Errors Report, *Oracle Purchasing User's Guide*

Receiving Open Interface

The Receiving Open Interface is used for processing and validating receipt data that comes from sources other than the Receipts window in Purchasing. These sources include the following:

- Receipt information from other Oracle Applications or your existing non-Oracle systems.
- Barcoded and other receiving information from scanners and radio frequency devices.
- Advance Shipment Notices (ASNs) sent from suppliers.

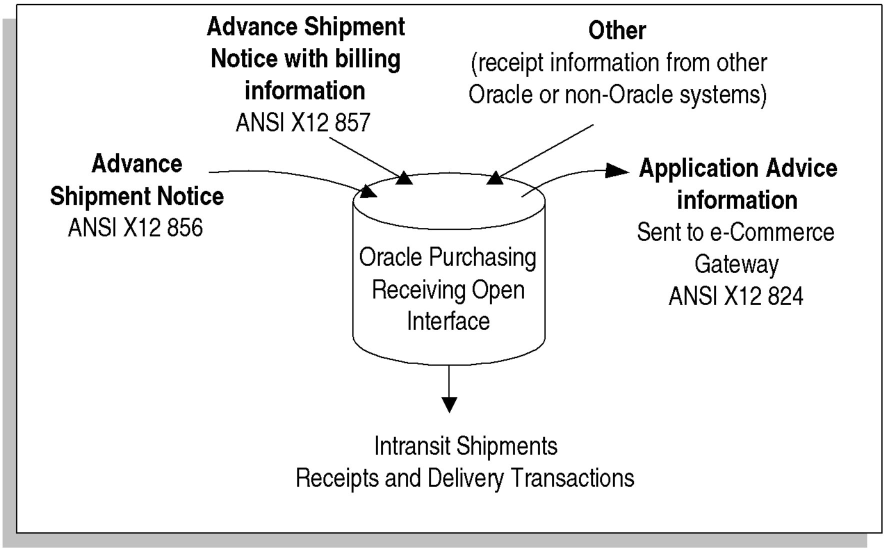
The Receiving Open Interface maintains the integrity of the new data as well as the receipt data already in Purchasing.

The Receiving Open Interface does not support lot or serial numbering, separate receive and deliver transactions (it supports receive transactions and direct receipts), corrections, returns, movement statistics, or dynamic locators.

The purpose of this essay is to explain how to use the Receiving Open Interface so that you can integrate other applications with Purchasing.

Functional Overview

Figure 8–4 Functional Overview


Advance Shipment Notice with billing information
ANSI X12 857
Other
(receipt information from other Oracle or non-Oracle systems)
Advance Shipment Notice
ANSI X12 856
Application Advice information
Sent to e-Commerce Gateway
ANSI X12 824
Oracle Purchasing Receiving Open Interface
Intransit Shipments
Receipts and Delivery Transactions

The diagram above shows the inputs and outputs that comprise the interface process.

Within the Receiving Open Interface, receipt data is validated for compatibility with Purchasing. There are two Receiving Open Interface tables:

- RCV_HEADERS_INTERFACE
- RCV_TRANSACTIONS_INTERFACE

EDI Transaction Types

The Electronic Data Interchange (EDI) transaction types supported by the Receiving Open Interface are as follows:

- Inbound Advance Shipment Notices (ANSI X12 856 or EDIFACT DESADV). These include Original (New), Cancellation, and Test ASNs.
- Inbound ASNs with billing information (ANSI X12 857). These also include Original (New), Cancellation, and Test ASNs.

- Outbound Application Advices (ANSI X12 824 or EDIFACT APERAK).

An ASN is transmitted through EDI from a supplier to let the receiving organization know 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*.

Validation and Overview

Receipt data that is entered through the Receipts window in Purchasing is derived, defaulted, and validated by the Receipts window. Most receipt data that is imported through the Receiving Open Interface is derived, defaulted, and validated by the receiving transaction pre-processor. The pre-processor is a program that the Receiving Transaction Processor initiates for data entered in the Receiving Open Interface. The pre-processor simulates, in Batch mode, what the receiving windows do when you save a transaction.

The following steps provide an overview of what the Receiving Open Interface does for each receipt:

Pre-processor Header-Level Validation

- You load the receipt data into the RCV_HEADERS_INTERFACE and RCV_TRANSACTIONS_INTERFACE tables, using EDI or your own program.
- The pre-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 pre-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 pre-processor defaults the correct TO_ORGANIZATION_ID.
- The pre-processor validates the receipt header information in the RCV_HEADERS_INTERFACE table to ensure the integrity of the information. For example, the SHIPPED_DATE should not be later than today. Only successfully validated header information is imported into the Purchasing tables.
- If no fatal errors are detected at the header level, the Receiving Transaction Processor selects all the lines in the RCV_TRANSACTIONS_INTERFACE table associated with each header and calls the pre-processor to perform line-level pre-processing.

Pre-processor Line-Level Validation

- The pre-processor derives and defaults any missing receipt line information in the RCV_TRANSACTIONS_INTERFACE table.
- The pre-processor validates the receipt line information to ensure the integrity of the information.
- For successfully validated lines, the pre-processor deletes the original RCV_TRANSACTIONS_INTERFACE line and creates the new, validated lines. Sometimes two or more validated rows are created in the RCV_TRANSACTIONS_INTERFACE table to correctly represent the original imported row.

Errors

If errors are detected in any of the above steps, 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 Purchasing. To view errors specifically for ASBNs, use the Purchasing Interface Errors Report.

Rows that fail validation in the Receiving Open Interface tables, producing errors, do not get imported into Purchasing (into the RCV_SHIPMENT_HEADERS, RCV_SHIPMENTS_LINES, and other applicable Purchasing tables). For example, an ASN can contain shipments from multiple purchase orders. If the purchase order number for one of the shipments is wrong, the shipment or the entire ASN will fail, depending on how the profile option *RCV: Fail All ASN Lines if One Line Fails* is set.

Receiving Transaction Processor Activities

After performing header- and line-level validation, the pre-processor checks the profile option *RCV: Fail All ASN Lines if One Line Fails.* If the profile option is set to 'Yes' and any line failed validation, the pre-processor fails the entire transaction. If the profile option is set to 'No' (and TEST_FLAG is not 'Y'), the Receiving Transaction Processor takes over and, for all successfully processed records, performs the same steps that occur when you normally save receipt information in Purchasing:

- Populates the RCV_SHIPMENT_HEADERS table in Purchasing with the receipt header information.
- Populates the RCV_SHIPMENT_LINES table in Purchasing for each receipt header entry in the RCV_SHIPMENT_HEADERS table in Purchasing.
- Populates the RCV_TRANSACTIONS table in 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 tables MTL_SUPPLY and RCV_ SUPPLY.
- 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.

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, it 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’s specified CUM quantity. If not, it 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

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 simply (for example) 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). 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 to determine which shipment lines to consume. 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 Purchasing, the Over Receipt Quantity Tolerance is 10%, 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 Purchasing rejects the last ASN line (or the whole ASN if the ASN has just one line) and creates an error in the PO_ INTERFACE_ERRORS table. Purchasing receives none of the units for those ASN lines that were rejected.

Purchasing does not require a Promised or Need-By date for an item that is unplanned; for unplanned items, 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, Purchasing skips that shipment and goes to the next.

Setting Up the Receiving Open Interface

You must complete the following setup steps in 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 User’s Guide*.

- In the Receiving Options window in Purchasing, select Warning, Reject, or None in the ASN Control field to determine how 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're 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 all processing, and as such it must be set up and running.

See also: Debugging on page 8-100.

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 you insert into the RCV_HEADERS_INTERFACE table, the Receiving Open Interface creates a shipment header; for each row 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.

When describing the table columns in the following graphics, the following definitions are used:

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 e-Commerce Gateway or a program 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.

Derived

The Receiving Open Interface is capable of deriving or defaulting columns in this category. If you provide your own value, the Receiving Open Interface uses it, if it is valid. If you leave the column blank, the Receiving Open Interface can derive it, based on other column values, if they're provided. For example, the column VENDOR_ID is defaulted in the RCV_HEADERS_INTERFACE table only if a value is provided in the VENDOR_NUM or VENDOR_NAME column. In general, the default values are defaulted in the same way that they are defaulted when you manually enter receipts in the Receipts, Receiving Transactions, or Manage Shipments windows in Purchasing.

Columns like those in the following example indicate that one of the pair can be derived if the other is provided:

Example Column Name	Type	Required	Derived	Optional
EXAMPLE_CODE	Varchar2		*conditionally*	
EXAMPLE_ID	Number			

Optional

You do not have to enter values for columns in this category.

Reserved for Future Use

The Receiving Open Interface does not support (validate) columns in this category as of this initial release. You should not populate values in these columns.

Receiving Headers Interface Table Description

The following graphic describes the receiving headers interface table.

Attention: A Derived column marked with an asterisk (x*) indicates that the Receiving Transaction Processor inserts values into these columns automatically, so you should not insert your own values.

Table 8–9 Receiving Open Interface Headers Table

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
HEADER_INTERFACE_ID	Number	x			
GROUP_ID	Number	x			
EDI_CONTROL_NUM	Varchar2			x	

Table 8–9   Receiving Open Interface Headers Table

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
PROCESSING_STATUS_CODE	Varchar2	x			
RECEIPT_SOURCE_CODE	Varchar2	x			
ASN_TYPE	Varchar2	*conditionally*			
TRANSACTION_TYPE	Varchar2	x			
AUTO_TRANSACT_CODE	Varchar2	*conditionally*			
TEST_FLAG	Varchar2			x	
LAST_UPDATE_DATE	Date	x			
LAST_UPDATED_BY	Number	x			
LAST_UPDATE_LOGIN	Number			x	
CREATION_DATE	Date	x			
CREATED_BY	Number	x			
NOTICE_CREATION_DATE	Date			x	
SHIPMENT_NUM	Varchar2	*conditionally*			
RECEIPT_NUM	Varchar2	*conditionally*			
RECEIPT_HEADER_ID	Number		*conditionally*		
VENDOR_NAME	Varchar2	x	*conditionally*		
VENDOR_NUM	Varchar2				
VENDOR_ID	Number				
VENDOR_SITE_CODE	Varchar2		*conditionally*	x	
VENDOR_SITE_ID	Number				
FROM_ORGANIZATION_ CODE	Varchar2				x
FROM_ORGANIZATION_ID	Number				
SHIP_TO_ORGANIZATION_ CODE	Varchar2	*conditionally*	*conditionally*		
SHIP_TO_ORGANIZATION_ID	Number				
LOCATION_CODE	Varchar2		*conditionally*	x	
LOCATION_ID	Number				
BILL_OF_LADING	Varchar2			x	

Table 8–9 Receiving Open Interface Headers Table

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
PACKING_SLIP	Varchar2			x	
SHIPPED_DATE	Date	*conditionally*			
FREIGHT_CARRIER_CODE	Varchar2			x	
EXPECTED_RECEIPT_DATE	Date			x	
RECEIVER_ID	Number				x
NUM_OF_CONTAINERS	Number			x	
WAYBILL_AIRBILL_NUM	Varchar2			x	
COMMENTS	Varchar2			x	
GROSS_WEIGHT	Number			x	
GROSS_WEIGHT_UOM_CODE	Varchar2			x	
NET_WEIGHT	Number			x	
NET_WEIGHT_UOM_CODE	Varchar2			x	
TAR_WEIGHT	Number			x	
TAR_WEIGHT_UOM_CODE	Varchar2			x	
PACKAGING_CODE	Varchar2			x	
CARRIER_METHOD	Varchar2			x	
CARRIER_EQUIPMENT	Varchar2			x	
SPECIAL_HANDLING_CODE	Varchar2			x	
HAZARD_CODE	Varchar2			x	
HAZARD_CLASS	Varchar2			x	
HAZARD_DESCRIPTION	Varchar2			x	
FREIGHT_TERMS	Varchar2			x	
FREIGHT_BILL_NUMBER	Varchar2			x	
INVOICE_NUM	Varchar2	*conditionally*			
INVOICE_DATE	Date	*conditionally*			
TOTAL_INVOICE_AMOUNT	Number	*conditionally*			
TAX_NAME	Varchar2			x	
TAX_AMOUNT	Number			x	

Table 8–9 Receiving Open Interface Headers Table

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
FREIGHT_AMOUNT	Number			x	
CURRENCY_CODE	Varchar2			x	
CONVERSION_RATE	Number			x	
CONVERSION_RATE_TYPE	Varchar2			x	
CONVERSION_RATE_DATE	Date			x	
PAYMENT_TERMS_NAME	Varchar2		*conditionally*	x	
PAYMENT_TERMS_ID	Number				
ATTRIBUTE_CATEGORY	Varchar2			x	
ATTRIBUTE1	Varchar2			x	
ATTRIBUTE2	Varchar2			x	
ATTRIBUTE3	Varchar2			x	
ATTRIBUTE4	Varchar2			x	
ATTRIBUTE5	Varchar2			x	
ATTRIBUTE6	Varchar2			x	
ATTRIBUTE7	Varchar2			x	
ATTRIBUTE8	Varchar2			x	
ATTRIBUTE9	Varchar2			x	
ATTRIBUTE10	Varchar2			x	
ATTRIBUTE11	Varchar2			x	
ATTRIBUTE12	Varchar2			x	
ATTRIBUTE13	Varchar2			x	
ATTRIBUTE14	Varchar2			x	
ATTRIBUTE15	Varchar2			x	
USSGL_TRANSACTION_CODE	Varchar2			x	
EMPLOYEE_NAME	Varchar2	*conditionally*	*conditionally*		
EMPLOYEE_ID	Number				
INVOICE_STATUS_CODE	Varchar2			x	
VALIDATION_FLAG	Varchar2	x			

Table 8–9 Receiving Open Interface Headers Table

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
REQUEST_ID	Number		x *		
PROCESSING_REQUEST_ID	Number		x *		

Receiving Transactions Interface Table Description

The following graphic describes the receiving transactions interface table.

Attention: A Derived column marked with an asterisk (x*) indicates that the Receiving Transaction Processor inserts values into these columns automatically, so you should not insert your own values.

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
INTERFACE_TRANSACTION_ID	Number	x			
GROUP_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
TRANSACTION_TYPE	Varchar2	x			
TRANSACTION_DATE	Date	x			
PROCESSING_STATUS_CODE	Varchar2	x			

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
PROCESSING_MODE_CODE	Varchar2	x			
PROCESSING_REQUEST_ID	Number		x *		
TRANSACTION_STATUS_CODE	Varchar2	x			
CATEGORY_ID	Number	*conditionally*	*conditionally*		
ITEM_CATEGORY	Varchar2				
QUANTITY	Number	x			
UNIT_OF_MEASURE	Varchar2	x			
INTERFACE_SOURCE_CODE	Varchar2			x	
INTERFACE_SOURCE_LINE_ID	Number				x
INV_TRANSACTION_ID	Number				x
ITEM_ID	Number	*conditionally*	*conditionally*		
ITEM_NUM	Varchar2				
ITEM_DESCRIPTION	Varchar2	x			
ITEM_REVISION	Varchar2	*conditionally*	*conditionally*		
UOM_CODE	Varchar2				x
EMPLOYEE_ID	Number	*conditionally*	*conditionally*		
AUTO_TRANSACT_CODE	Varchar2	x			
SHIPMENT_HEADER_ID	Number				x
SHIPMENT_LINE_ID	Number				x
SHIP_TO_LOCATION_ID	Number	*conditionally*	*conditionally*		
SHIP_TO_LOCATION_CODE	Varchar2				
PRIMARY_QUANTITY	Number				x
PRIMARY_UNIT_OF_MEASURE	Varchar2				x
RECEIPT_SOURCE_CODE	Varchar2	x			
VENDOR_ID	Number	x	*conditionally*		
VENDOR_NUM	Varchar2				
VENDOR_NAME	Varchar2				

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
VENDOR_SITE_ID	Number		x	x	
VENDOR_SITE_CODE	Varchar2				
FROM_ORGANIZATION_ID	Number				x
TO_ORGANIZATION_CODE	Varchar2	*conditionally*	*conditionally*		
TO_ORGANIZATION_ID	Number				
ROUTING_HEADER_ID	Number				x
ROUTING_STEP_ID	Number				x
SOURCE_DOCUMENT_CODE	Varchar2	x			
PARENT_TRANSACTION_ID	Number				x
PO_HEADER_ID	Number	x	*conditionally*		
DOCUMENT_NUM	Varchar2				
PO_REVISION_NUM	Number			x	
PO_RELEASE_ID	Number		*conditionally*	x	
RELEASE_NUM	Number				
PO_LINE_ID	Number	*conditionally*	*conditionally*		
DOCUMENT_LINE_NUM	Number				
PO_LINE_LOCATION_ID	Number		*conditionally*	x	
DOCUMENT_SHIPMENT_LINE_NUM	Number				
PO_UNIT_PRICE	Number			x	
CURRENCY_CODE	Varchar2			x	
CURRENCY_CONVERSION_TYPE	Varchar2			x	
CURRENCY_CONVERSION_RATE	Number			x	
CURRENCY_CONVERSION_DATE	Date			x	
PO_DISTRIBUTION_ID	Number		*conditionally*	x	
DOCUMENT_DISTRIBUTION_NUM	Number				

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
REQUISITION_LINE_ID	Number				x
REQ_DISTRIBUTION_ID	Number				x
CHARGE_ACCOUNT_ID	Number				x
SUBSTITUTE_UNORDERED_ CODE	Varchar2				x
RECEIPT_EXCEPTION_FLAG	Varchar2				x
ACCRUAL_STATUS_CODE	Varchar2				x
INSPECTION_STATUS_CODE	Varchar2				x
INSPECTION_QUALITY_CODE	Varchar2				x
DESTINATION_TYPE_CODE	Varchar2		*conditionally*	x	
SUBINVENTORY	Varchar2	*conditionally*	*conditionally*		
WIP_ENTITY_ID	Number				x
WIP_LINE_ID	Number				x
DEPARTMENT_CODE	Varchar2				x
WIP_REPETITIVE_SCHEDULE_ ID	Number				x
WIP_OPERATION_SEQ_NUM	Number				x
WIP_RESOURCE_SEQ_NUM	Number				x
BOM_RESOURCE_ID	Number				x
SHIPMENT_NUM	Varchar2				x
FREIGHT_CARRIER_CODE	Varchar2		*conditionally*		
BILL_OF_LADING	Varchar2		*conditionally*		
PACKING_SLIP	Varchar2			x	
SHIPPED_DATE	Date				x
EXPECTED_RECEIPT_DATE	Date	*conditionally*	*conditionally*		
ACTUAL_COST	Number			x	
TRANSFER_COST	Number			x	
TRANSPORTATION_COST	Number			x	

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
TRANSPORTATION_ACCOUNT_ ID	Number			x	
NUM_OF_CONTAINERS	Number			x	
WAYBILL_AIRBILL_NUM	Varchar2			x	
VENDOR_ITEM_NUM	Varchar2	*conditionally*	*conditionally*		
VENDOR_LOT_NUM	Varchar2				x
RMA_REFERENCE	Varchar2			x	
COMMENTS	Varchar2			x	
ATTRIBUTE_CATEGORY	Varchar2			x	
ATTRIBUTE1	Varchar2			x	
ATTRIBUTE2	Varchar2			x	
ATTRIBUTE3	Varchar2			x	
ATTRIBUTE4	Varchar2			x	
ATTRIBUTE5	Varchar2			x	
ATTRIBUTE6	Varchar2			x	
ATTRIBUTE7	Varchar2			x	
ATTRIBUTE8	Varchar2			x	
ATTRIBUTE9	Varchar2			x	
ATTRIBUTE10	Varchar2			x	
ATTRIBUTE11	Varchar2			x	
ATTRIBUTE12	Varchar2			x	
ATTRIBUTE13	Varchar2			x	
ATTRIBUTE14	Varchar2			x	
ATTRIBUTE15	Varchar2			x	
SHIP_HEAD_ATTRIBUTE_ CATEGORY	Varchar2				x
SHIP_HEAD_ATTRIBUTE1	Varchar2				x
SHIP_HEAD_ATTRIBUTE2	Varchar2				x
SHIP_HEAD_ATTRIBUTE3	Varchar2				x

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
SHIP_HEAD_ATTRIBUTE4	Varchar2				x
SHIP_HEAD_ATTRIBUTE5	Varchar2				x
SHIP_HEAD_ATTRIBUTE6	Varchar2				x
SHIP_HEAD_ATTRIBUTE7	Varchar2				x
SHIP_HEAD_ATTRIBUTE8	Varchar2				x
SHIP_HEAD_ATTRIBUTE9	Varchar2				x
SHIP_HEAD_ATTRIBUTE10	Varchar2				x
SHIP_HEAD_ATTRIBUTE11	Varchar2				x
SHIP_HEAD_ATTRIBUTE12	Varchar2				x
SHIP_HEAD_ATTRIBUTE13	Varchar2				x
SHIP_HEAD_ATTRIBUTE14	Varchar2				x
SHIP_HEAD_ATTRIBUTE15	Varchar2				x
SHIP_LINE_ATTRIBUTE_ CATEGORY	Varchar2				x
SHIP_LINE_ATTRIBUTE1	Varchar2				x
SHIP_LINE_ATTRIBUTE2	Varchar2				x
SHIP_LINE_ATTRIBUTE3	Varchar2				x
SHIP_LINE_ATTRIBUTE4	Varchar2				x
SHIP_LINE_ATTRIBUTE5	Varchar2				x
SHIP_LINE_ATTRIBUTE6	Varchar2				x
SHIP_LINE_ATTRIBUTE7	Varchar2				x
SHIP_LINE_ATTRIBUTE8	Varchar2				x
SHIP_LINE_ATTRIBUTE9	Varchar2				x
SHIP_LINE_ATTRIBUTE10	Varchar2				x
SHIP_LINE_ATTRIBUTE11	Varchar2				x
SHIP_LINE_ATTRIBUTE12	Varchar2				x
SHIP_LINE_ATTRIBUTE13	Varchar2				x
SHIP_LINE_ATTRIBUTE14	Varchar2				x

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
SHIP_LINE_ATTRIBUTE15	Varchar2				x
USSGL_TRANSACTION_CODE	Varchar2				x
GOVERNMENT_CONTEXT	Varchar2				x
REASON_ID	Number			x	
DESTINATION_CONTEXT	Varchar2				x
SOURCE_DOC_QUANTITY	Number				x
SOURCE_DOC_UNIT_OF_ MEASURE	Varchar2				x
FROM_SUBINVENTORY	Varchar2				x
INTRANSIT_OWNING_ORG_ID	Number				x
MOVEMENT_ID	Number				x
USE_MTL_LOT	Number				x
USE_MTL_SERIAL	Number				x
TAX_NAME	Varchar2			x	
TAX_AMOUNT	Number			x	
NOTICE_UNIT_PRICE	Number			x	
HEADER_INTERFACE_ID	Number	x			
VENDOR_CUM_SHIPPED_ QUANTITY	Number			x	
TRUCK_NUM	Varchar2			x	
CONTAINER_NUM	Varchar2			x	
LOCATION_CODE	Varchar2		*conditionally*	x	
LOCATION_ID	Number				
FROM_ORGANIZATION_CODE	Varchar2				x
INTRANSIT_OWNING_ORG_ CODE	Varchar2				x
ROUTING_CODE	Varchar2				x
ROUTING_STEP	Varchar2				x

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
DELIVER_TO_PERSON_NAME	Varchar2	*conditionally*	*conditionally*		
DELIVER_TO_PERSON_ID	Number				
DELIVER_TO_LOCATION_CODE	Varchar2	*conditionally*	*conditionally*		
DELIVER_TO_LOCATION_ID	Number				
LOCATOR	Varchar2	*conditionally*	*conditionally*		
LOCATOR_ID	Number				
REASON_NAME	Varchar2			x	
VALIDATION_FLAG	Varchar2	x			
SUBSTITUTE_ITEM_ID	Number		*conditionally*	x	
SUBSTITUTE_ITEM_NUM	Varchar2				
QUANTITY_SHIPPED	Number				x
QUANTITY_INVOICED	Number				x
REQ_NUM	Varchar2				x
REQ_LINE_NUM	Number				x
REQ_DISTRIBUTION_NUM	Number				x
WIP_ENTITY_NAME	Varchar2				x
WIP_LINE_CODE	Varchar2				x
RESOURCE_CODE	Varchar2				x
SHIPMENT_LINE_STATUS_ CODE	Varchar2		x *		
BARCODE_LABEL	Varchar2			x	
TRANSFER_PERCENTAGE	Number				x
QA_COLLECTION_ID	Number				x
COUNTRY_OF_ORIGIN_CODE	Varchar2			x	
OE_ORDER_HEADER_ID	Number				x
OE_ORDER_LINE_ID	Number				x
CUSTOMER_ID	Number				x
CUSTOMER_SITE_ID	Number				x

Table 8–10 Receiving Open Interface Transactions Table

RCV_TRANSACTIONS_ INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
CUSTOMER_ITEM_NUM	Varchar2				x

Required Data for RCV_HEADERS_INTERFACE

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

HEADER_INTERFACE_ID

Purchasing provides a unique-sequence generator to generate a unique identifier for this column. If you're importing data through e-Commerce Gateway, a value is provided automatically.

GROUP_ID

Purchasing provides a group identifier for a set of transactions that should be processed together.

PROCESSING_STATUS_CODE

This columns 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 only of 'VENDOR'.

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 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 the header record was created. CREATED_BY indicates the loading program or user ID that was used to import the header record. If you're importing data through e-Commerce Gateway, values are provided in these columns automatically.

VENDOR_NAME, VENDOR_NUM, or VENDOR_ID

VENDOR_NAME and VENDOR_NUM indicate the supplier name and number for the shipment. Both must be a valid name or number in 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.

VALIDATION_FLAG

This column indicates whether to validate a row before processing it. It accepts values of 'Y' or 'N'. The Receiving Open Interface provides a default value of 'Y'.

Conditionally Required Data for RCV_HEADERS_INTERFACE

Additionally, you may have to enter values for the following conditionally required columns in the RCV_HEADERS_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.

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 in this column must be unique from the supplier for a period of one year.

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.

SHIP_TO_ORGANIZATION_CODE or SHIP_TO_ORGANIZATION_ID

These columns indicate the destination organization for the shipment. A valid inventory organization code in Purchasing is required for an ASN. If the supplier does not know the ship-to organization, then it 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 it 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.

SHIPPED_DATE

This column indicates 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. It must also be earlier than or equal to the EXPECTED_RECEIPT_DATE.

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.

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 Purchasing or Oracle Applications.

Required Data for RCV_TRANSACTIONS_INTERFACE

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

INTERFACE_TRANSACTION_ID

Purchasing provides a unique-sequence generator to generate a unique identifier for the receiving transaction line. If you're importing data through e-Commerce Gateway, a value is provided automatically.

GROUP_ID

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.

LAST_UPDATE_DATE, LAST_UPDATED_BY, CREATION_DATE, and CREATED_BY

LAST_UPDATE_DATE indicates 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 the line was created. CREATED_BY indicates the loading program or user ID that was used to import the line. If you're importing data through e-Commerce Gateway, values are provided in these columns automatically.

TRANSACTION_TYPE

This column indicates the transaction purpose code. It accepts values of 'SHIP' for a standard shipment (an ASN or ASBN) or 'RECEIVE' for a standard receipt.

TRANSACTION_DATE

This column indicates the date of the transaction. The date must be in an open Purchasing and General Ledger period and, if Inventory is installed, also be in an open 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'.

QUANTITY

This column indicates the shipment quantity. The value in this column must be a positive number.

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.

UNIT_OF_MEASURE

This column indicates the shipment quantity unit of measure (UOM). If the UOM is different from the primary UOM defined in Purchasing and/or the source document UOM, then a conversion must be defined between the two UOMs. Navigate to the Unit of Measure Conversions window by choosing Setup > Units of Measure > Conversions.

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.

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 the other values. 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 you are receiving against.

ITEM_NUM indicates the Purchasing item number of the item you are receiving. The item number must be defined in Purchasing for the DOCUMENT_NUM provided and the SHIP_TO_ORGANIZATION_CODE.

VENDOR_ITEM_NUM indicates the supplier item number of the item you are receiving. The value in this column must be defined in Purchasing as a supplier item number on the specified purchase order.

AUTO_TRANSACT_CODE

This column 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 or not 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 below shows the combinations of TRANSACTION_TYPE and AUTO_TRANSACT_CODE values you can choose in the RCV_TRANSACTIONS_INTERFACE table to create an ASN or ASBN shipment header and shipment line(s), a receiving transaction, or a receiving and delivery transaction.

	AUTO_TRANSACT_CODE		
TRANSACTION_TYPE	**NULL**	**RECEIVE**	**DELIVER**
SHIP	Shipment header and shipment line(s) created	Receiving transaction created	Receiving and delivery transaction created
RECEIVE	Receiving transaction created	Receiving transaction created	Receiving and delivery transaction created

RECEIPT_SOURCE_CODE

This column indicates the source of the shipment. It accepts a value of 'VENDOR' only. The Receiving Open Interface can derive the value here if one is provided in the RCV_HEADERS_INTERFACE table.

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.

SOURCE_DOCUMENT_CODE

This column indicates the document type for the shipment. It accepts a value of ’PO’ only.

DOCUMENT_NUM or PO_HEADER_ID

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 Purchasing. If you provide a value in either the DOCUMENT_NUM or PO_HEADER_ID column, the other can be derived.

HEADER_INTERFACE_ID

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’re importing data through e-Commerce Gateway, a value is provided automatically.

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’.

Conditionally Required Data for RCV_TRANSACTIONS_INTERFACE

Additionally, you may have to enter values for the following conditionally required columns in the RCV_TRANSACTIONS_INTERFACE table:

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 don’t 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_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 Purchasing) for the item you’re 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.

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.

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 need 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.

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. You can have different ship-to organizations specified for different lines, if no SHIP_TO_ORGANIZATION_CODE is provided in the RCV_HEADERS_INTERFACE table.

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.

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.

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 can't 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).

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.

Derived Data

In general, the Receiving Open Interface derives or defaults derived columns using logic similar to that used by the Receipts, Receiving Transactions, or Manage Shipments windows. Purchasing never overrides information that you provide in derived columns.

In general, 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 Purchasing. For example:

- 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 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 Purchasing tables.
- SUBSTITUTE_ITEM_NUM - The value in this column must be defined in 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 a 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 column VENDOR_CUM_SHIPPED_QUANTITY does not match what you have received, then your supplier is notified through an Application Advice (if you're receiving ASNs through e-Commerce Gateway).

Validation

The Receiving Open Interface does not perform any validations for columns that are indicated as ”Reserved for Future Use” on the previous pages.

Standard Validation

Oracle Purchasing validates all required columns in the interface tables. For specific information on the data implied by these columns, see your *Oracle Purchasing Technical Reference Manual, Release 11i* for details.

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 to do a test run of the Receiving Open Interface, see and fix the errors, and run the program again.

To debug the receiving transaction pre-processor:

1. Set the profile option *PO: Enable Sql Trace for Receiving Processor* to Yes.

   Setting this profile option to Yes provides more detailed error information in the View Log screen of the Submit Request window when you run the Receiving Transaction Processor in step 3. (’Yes’ also places in the database a trace file, which can be used by Oracle Support Services if needed.)

2. Load your receiving data into the Receiving Open Interface tables using e-Commerce Gateway or other means.

3. Navigate to the Submit Request window by choosing Reports > Run and submit the Receiving Transaction Processor.

4. When the Receiving Transaction Processor completes, choose the View Log button to see what errors occurred, if any.

   Because the profile option in step 1 was set to Yes, the View Log screen shows the pre-processor’s actions as it processed the data, from start to finish. If an error occurs during this process, you see not just the error, but where in the process the error occurred.

5. Check that derived and defaulted data was derived and defaulted correctly.

   You can use SQL*Plus to do this if you’re on a test environment, or use the Transaction Statuses window and the Help > Diagnostics > Examine menu to check the values.

   For example, if you provided a DOCUMENT_NUM in the RCV_TRANSACTIONS_INTERFACE table but no PO_HEADER_ID, the Receiving Open Interface should derive the correct PO_HEADER_ID for you. Derived and defaulted data is shown in the Receiving Open Interface table descriptions on the previous pages.

6. Run the Receiving Interface Errors Report if you want to see a list only of the errors that occurred.

   The View Log screen displays errors in the context in which they occurred. The Receiving Interface Errors Report shows you just the errors.

7. Make the necessary fixes for the errors or incorrectly defaulted data, if any.

8. Repeat steps 2 through 7 until you have successfully processed the data with no errors.

See Also

Receiving Transaction Processor, *Oracle Purchasing User's Guide*

Resolving Failed Receiving Open Interface Rows

Error Messages

Oracle Purchasing may display specific error messages during interface processing. For more details on these messages, please see the *Oracle Applications Messages Manual*, in HTML format on the documentation CD-ROM for Release 11*i*.

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.)

See Also

Receiving Interface Errors Report, *Oracle Purchasing User's Guide*

Purchasing Interface Errors Report, *Oracle Purchasing User's Guide*

9

Oracle Quality Open Interfaces

This chapter contains information about the following Oracle Quality open interfaces:

- Collection Import Interface on page 9-2
- Collection Plan Views on page 9-17

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.“

ATTENTION: 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.

Attention: Do not confuse this step with running the Collection Import Manager in the "Update Transaction" mode.

See Also

Collection Import Interface Table on page 9-3

Example: Collection Import SQL Script on page 9-11

Collection Import Manager on page 9-15

Updating Collection Import, *Oracle Quality User's Guid*e

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.

Table 9–1 Collection Import Interface

[Collection Import Interface Table]

Column Name	Data Type	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
CHARACTER1 through CHARACTER100	Varchar(150)					x
COLLECTION_ID	Number(38)			x		
COMP_GEN_LOC_CTRL_CODE	Number		x[1]			
COMP_ITEM	Varchar2(2000)					x
COMP_ITEM_ID	Number		x			x
COMP_LOCATION_CONTROL_CODE	Number		x[1]			
COMP_LOCATOR	Varchar2(2000)					x
COMP_LOCATOR_ID	Number		x			x
COMP_LOT_NUMBER	Varchar2(30)					x
COMP_RESTRICT_LOCATORS_CODE	Number		x[1]			
COMP_RESTRICT_SUBINV_CODE	Number		x[1]			
COMP_REVISION	Varchar2(3)					x
COMP_REVISION_QTY_CONTROL_CODE	Number		x[1]			
COMP_SERIAL_NUMBER	Varchar2(30)					x
COMP_SUB_LOCATOR_TYPE	Number		x[1]			
COMP_SUBINVENTORY	Varchar2(10)					x
COMP_UOM	Varchar2(3)					x
CREATED_BY	Number		x[1]			
CREATION_DATE	Date		x[1]			
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		x[1]			x
GROUP_ID	Number		x[1]			

Table 9–1 Collection Import Interface

[Collection Import Interface Table]						
Column Name	**Data Type**	**Required**	**Derived (Leave Null)**	**Derived or User Optional**	**Optional**	**Plan Specific**
INSERT_TYPE	Number				x	
ITEM	Varchar2(2000)					x
ITEM_ID	Number		x			x
JOB_NAME	Varchar2(240)					x
LAST_UPDATE_DATE	Date		x			
LAST_UPDATE_LOGIN	Number		x[1]			
LAST_UPDATED_BY	Number		x[1]			
LINE_ID	Number		x			x
LOCATION_CONTROL_CODE	Number		x[1]			x
LOCATOR	Varchar2(2000)					x
LOCATOR_ID	Number		x			x
LOT_NUMBER	Varchar2(30)					x
MARKER	Number		x[1]			
MATCHING ELEMENTS	Varchar2(1000)				x	
ORGANIZATION_CODE	Varchar2(3)	x				
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

Table 9–1 Collection Import Interface

[Collection Import Interface Table]

Column Name	Data Type	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
PROCESS_STATUS	Number	x				
PRODUCTION_LINE	Varchar2(10)					x
PROGRAM_APPLICATION_ID	Number		x[1]			
PROGRAM_ID	Number		x[1]			
PROGRAM_UPDATE_DATE	Date		x[1]			
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		x[1]			
RESOURCE_CODE	Varchar2(10)					x
RESOURCE_ID	Number		x			x
RESTRICT_LOCATORS_CODE	Number		x[1]			x
RESTRICT_SUBINV_CODE	Number		x[1]			x
REVISION	Varchar2(3)					x
REVISION_QTY_CONTROL_CODE	Number		x[1]			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	

Table 9–1 Collection Import Interface

[Collection Import Interface Table]						
Column Name	Data Type	Required	Derived (Leave Null)	Derived or User Optional	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		x[1]			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		
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.

ATTENTION: 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.

ATTENTION: 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.

Attention: 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 on page 9-15

Creating Collection Plans, *Oracle Quality User's Guid*e

Collection Plan and Import Results Database Views, *Oracle Quality User's Guid*e

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)
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, on page 9-9).
- 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 on page 9-3

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)
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

10

Oracle Work in Process Open Interfaces

This chapter contains information about the following Oracle Work in Process open interfaces:

- Open Move Transaction Interface on page 10-2
- Open Resource Transaction Interface on page 10-17
- Work Order Interface on page 10-25

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

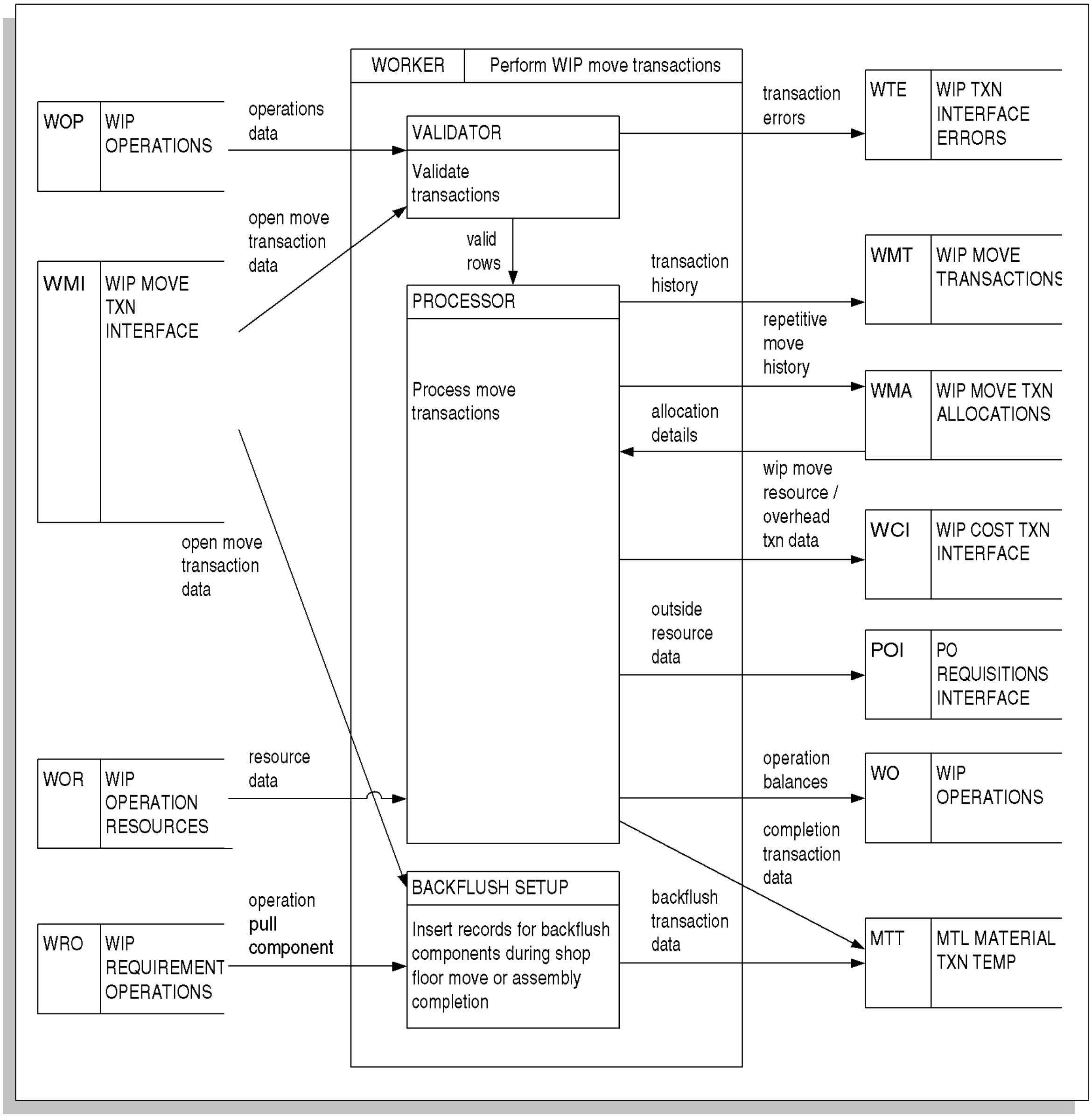
ATTENTION: You cannot add ad hoc operations through this interface even if the *WIP Allow Creation of New Operations* parameter is set.

The following information describes how you can use the Move Transaction Interface to integrate other applications with Oracle Work in Process.

Functional Overview

The following data flow diagram shows the key tables and programs that comprise the Move Transaction Interface:

Figure 10–1 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

The following describes the WIP_MOVE_TXN_INTERFACE table:

<table>
<tr><th colspan="2">Legend</th></tr>
<tr><td colspan="2"><b>Note:</b> in the following table, the numbers under the Required, Derived if Null, and Optional columns indicate that the referenced column is used for the following:</td></tr>
<tr><th>Type Number</th><th>Transaction Type Description</th></tr>
<tr><td>1</td><td>Move transaction</td></tr>
<tr><td>2</td><td>Move Completion (defined by TRANSACTION_TYPE = 2 and valid FM_OPERATION_SEQ_NUM)</td></tr>
<tr><td>3</td><td>Move Return (defined by TRANSACTION_TYPE = 2 and valid T0_OPERATION_SEQ_NUM)</td></tr>
</table>

Table 10–1 Move Transaction Interface

WIP_MOVE_TXN_INTERFACE					
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			
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			
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		

Table 10–1 Move Transaction Interface

WIP_MOVE_TXN_INTERFACE					
Column Name	Type	Required	Derived	Optional	Derived if Null
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		
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			
TRANSACTION_DATE	Date	1,2,3			
TRANSACTION_ID	Number		1,2,3		
TRANSACTION_QUANTITY	Number	1,2,3			
TRANSACTION_TYPE	Number			1,2,3	

Table 10–1 Move Transaction Interface

WIP_MOVE_TXN_INTERFACE					
Column Name	**Type**	**Required**	**Derived**	**Optional**	**Derived if Null**
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:

  1 Move Validation

  2 Move Processing

  3 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).

  1 Pending

  2 Running

  3 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:

  1 Move

  2 Move Completion

  3 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 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 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:

  1 Move Validation

  2 Move Processing

  3 Operation Backflush Setup

- The column PROCESS_STATUS contains the state of the transaction. You should always load 1 (Pending):

  1 Pending

  2 Running

  3 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:

Table 10–2 Completion Cost Calculation 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	
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 window's *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

ATTENTION: 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.

The following information describes how you can use the Resource Transaction Interface to integrate other applications with Oracle Work in Process.

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:

Table 10–3 WIP Cost Transaction Interface

[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

Table 10–3 WIP Cost Transaction Interface

[WIP_COST_TXN_INTERFACE]				
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	
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	

Table 10–3 WIP Cost Transaction Interface

[WIP_COST_TXN_INTERFACE]				
Column Name	**Type**	**Required**	**Derived**	**Optional**
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
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	
REQUEST_ID	Number		x	
RESOURCE_CODE	VarChar2(10)		x	x
RESOURCE_ID	Number		x	
RESOURCE_SEQ_NUM	Number	x		

Table 10–3 WIP Cost Transaction Interface

[WIP_COST_TXN_INTERFACE]				
Column Name	**Type**	**Required**	**Derived**	**Optional**
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		

ATTENTION: 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).

1 Resource Validation

2 Resource Processing

- The column PROCESS_STATUS contains the state of the transaction. You should always load 1 (Pending):

  1 Pending

  2 Running

  3 Error

- The column RESOURCE_ID column must be left NULL.
- You set TRANSACTION_TYPE to:

  1 for normal resource transactions

  2 for overhead transactions

  3 for 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 employee’s 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 organization's 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, and
- 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 of the Work Order Interface are:

- **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 customer's 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:

Table 10–4 Features of the Work Order Interface

	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		

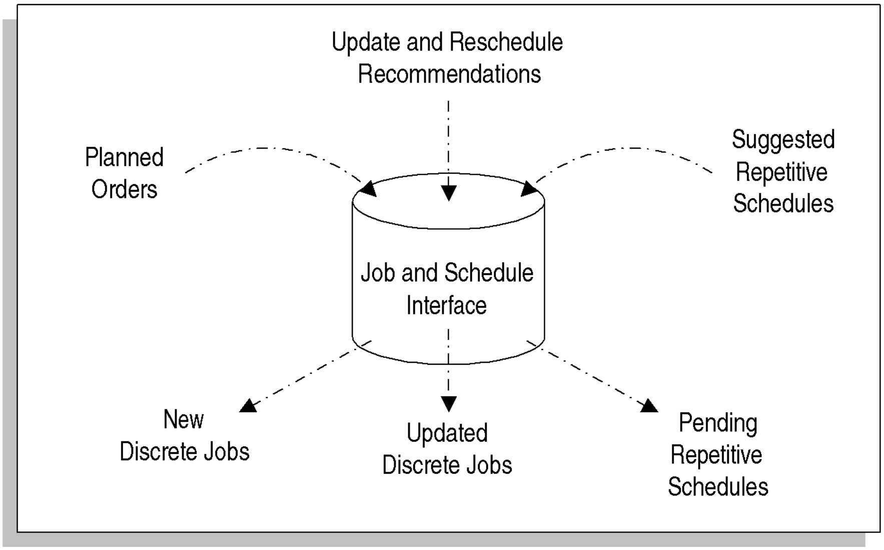
See Also

Overview of Reports and Programs, *Oracle Applications User's Guide*

Functional Overview

Figure 10-2 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.

Figure 10–2 Work Order Interface Input and Output


Update and Reschedule Recommendations
Planned Orders
Suggested Repetitive Schedules
Job and Schedule Interface
New Discrete Jobs
Updated Discrete Jobs
Pending Repetitive Schedules

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 Oracle Master Scheduling/MRP's 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, *Oracle Master Scheduling/MRP User's Guide, R11i).*

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

See Also

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.

See Also

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 job's 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).

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 or change operations, and 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.
- **Updating both header and detail information:** load the data into both the WIP_JOB_SCHEDULE INTERFACE and WIP_JOB_DTLS_INTERFACE tables.

The following sections describe the columns in each of the two Work Order 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:

Table 10–5 Work Order Interface: WIP_JOB_SCHEDULE_INTERFACETable

Legend

Note: The numbers under the Required, Optional/Derived if Null, Optional, and Derived or Ignored columns indicate that the column is used to do the following:

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

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table						
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 not defined for this assembly or reference, or if routing type or assembly type is 1, issues an error. Issues a warning if NOT NULL (required).
ALTERNATE_ ROUTING_ DESIGNATOR	Varchar2 (10)			1,4	2,3	See ALTERNATE_BOM_ DESIGNATOR.
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table						
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.
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.
CREATION_DATE	Date	1,2,3,4				Has a NOT NULL restriction; not loaded into the table; default is SYSDATE.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table

Column	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored	Additional Information
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.
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.
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table						
Column	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored	Additional Information
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.
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).
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table

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.
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table

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.
LOT_NUMBER	Varchar2 (30)		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.
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table						
Column	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored	Additional Information
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.
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.
PROCESS_PHASE	Number	1,2,3,4				See Control Columns.
PROCESS_STATUS	Number	1,2,3,4				See Control Columns.
PROCESS_TYPE						This column has a NOT NULL restriction. Must be set to running, error, or completed, otherwise error will occur (pending not used).

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table

Column	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored	Additional Information
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.
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table						
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.
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.
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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table

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 2 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.

Work Order Interface: WIP_JOB_SCHEDULE_INTERFACE Table

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	VarChar 2 (100)	3			1,2,4	See WIP_ENTITY_ID.
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.

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 Technical Reference Manual, R11i.

- 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/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, 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

> **Attention:** 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.

> **Attention:** Records with new tasks will only successfully load if those tasks have been associated with the job's 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 Pending.

NOTE: If you have installed Oracle Manufacturing Scheduling, and have its Constraint-Based Scheduling engine turned on, Unreleased is 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 table’s 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 for loading a resource
- 2 for loading a component
- 3 for loading an operation
- 4 for loading multiple resource 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, RESOURCE_SEQ_NUM, and WIP_ENTITY_ID. 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.

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table

Column Name	Type	Operation		Component		Resource		Schedule	Additional Information
		Add	Chg	Add	Chg	Add	Chg		
ACTIVITY_ID	Number	Null	Null	Null	Null	Opt	Opt	Null	Identifier for the activity
APPLIED_ RESOURCE_UNITS	Number	Null	Null	Null	Null	Opt	Opt	Null	Number of resource units charged.

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table									
Column Name	Type	Operation Add	Chg	Component Add	Chg	Resource Add	Chg	Schedule	Additional Information
APPLIED_ RESOURCE_VALUE	Number	Null	Null	Null	Null	Opt	Opt	Null	Value of the resource units charged.
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.
COMPLETION_ DATE	Date	Null	Null	Null	Null	Reqd	Opt	Null	Resource’s 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_SEQ_ ID	Number	Null	Reqd	Reqd	Null	Null	Null	Null	Identifier for the bill of materials component sequence.

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table

Column Name	Type	Operation		Component		Resource		Schedule	Additional Information
		Add	Chg	Add	Chg	Add	Chg		
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	
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.
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).

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table

Column Name	Type	Operation		Component		Resource		Schedule	Additional Information
		Add	Chg	Add	Chg	Add	Chg		
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.
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.
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
MINIMUM_ TRANSFER_ QUANTITY	Number	Reqd	Opt	Null	Null	Null	Null	Null	Required for loading operations optional when updating.

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table

Column Name	Type	Operation		Component		Resource		Schedule	Additional Information
		Add	Chg	Add	Chg	Add	Chg		
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.
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.
ORGANIZATION_ID	Number	Null	Null	Null	Null	Null	Null	Null	Identifier for the organization.
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.

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table									
Column Name	Type	Operation Add	Operation Chg	Component Add	Component Chg	Resource Add	Resource Chg	Schedule	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.
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.

Work Order Interface: WIP_JOB_ DTLS_INTERFACE Table

Column Name	Type	Operation		Component		Resource		Schedule	Additional Information
		Add	Chg	Add	Chg	Add	Chg		
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.
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.

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.

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 WIPconstraints. 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.

See Also

Oracle Work in Process Technical Reference Manual:

Processing Pending Jobs and Schedules, *Oracle Work in Process User’s Guide*

Index