Oracle® Manufacturing, Distribution, Sales and Service Open Interfaces Manual

Release 11

March 1998

ORACLE®

Enabling the Information Age™

Oracle® Manufacturing, Distribution, Sales and Service Open Interfaces Manual
Release 11

The part number for this book is A57322–01.

Major Contributors: Louis Bryan, Bryan Dobson, Sharon Goetz, Rachel Haas, Kevin Hamant, Ann Huybrechts, Susan Ramage, Sue Saperstein

Contents

Preface

This *Oracle® Manufacturing, Distribution, Sales and Service Open Interfaces Manual, Release 11* 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 11. 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 Bills of Material open interfaces.
- Chapter 3 contains information about Oracle Inventory open interfaces.
- Chapter 4 contains information about Oracle Master Scheduling/MRP and Oracle Supply Chain Planning open interfaces.
- Chapter 5 contains information about Oracle Order Entry/Shipping open interfaces.
- Chapter 6 contains information about Oracle Purchasing or Oracle Public Sector Purchasing open interfaces.
- Chapter 7 contains information about Oracle Quality open interfaces.
- Chapter 8 contains information about Oracle Service open interfaces.
- Chapter 9 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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 11.

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 Education Services at any one of our many Education Centers, or you can arrange for our trainers to teach at your facility. In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization 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 45 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 implemenation 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**.

CHAPTER

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.

Overview of Oracle Manufacturing, Distribution, Sales and Service Open Interfaces

Oracle Manufacturing, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service product 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, Distribution, Sales and Service products.
- Review, update, and resubmit imported data that failed validation.

- Export data from Oracle Manufacturing, Distribution, Sales and Service products

Oracle Manufacturing Interfaces

Open Interface Architectures

Oracle Manufacturing, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Marketing Open Interfaces Manual*
TRM	The tables, views, or modules are described in the product's Technical Reference Manual

Interface Name	Data Flow Direction	Table, View, or Process	Iface Man	TRM	Table, View, or Module 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
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
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

Table 1 – 1 Oracle Manufacturing Interfaces (Page 1 of 3)

Interface Name	Data Flow Direction	Table, View, or Process	Iface Man	TRM	Table, View, or Module Name
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	Inbound	Table	Yes	Yes	MRP_FORECAST_INTERFACE
Forecast Entries	Inbound	Process PL/SQL Table	Yes	Yes	T_FORECAST_INTERFACE T_FORECAST_DESIGNATOR
Master Schedule	Inbound	Table	Yes	Yes	MRP_SCHEDULE_INTERFACE
Master Schedule Relief	Inbound	Table		Yes	MRP_RELIEF_INTERFACE
Planner Workbench	Outbound	Process		Yes	Stored Procedure MRPPL06, or WIP_JOB_SCHEDULE_INTERFACE PO_REQUISITIONS_INTERFACE PO_RESCHEDULE_INTERFACE
Projected Requirements	Outbound	Table		Yes	MRP_RECOMMENDATIONS
Projected Supply	Outbound	Table		Yes	MRP_GROSS_REQUIREMENTS
Oracle Order Entry/Shipping					
Order Import	Inbound	Table	Yes	Yes	SO_HEADERS_INTERFACE_ALL SO_HEADER_ATTRIBUTES_INTERFACE SO_LINES_INTERFACE_ALL SO_LINE_ATTRIBUTES_INTERFACE SO_LINE_DETAILS_INTERFACE SO_SALES_CREDITS_INTERFACE SO_PRICE_ADJUSTMENTS_INTERFACE
Delivery–based Ship Confirm Open Interface	Inbound	Table	Yes	Yes	WSH_DELIVERIES_INTERFACE WSH_PACKED_CONTAINER_INTERFACE WSH_PICKING_DETAILS_INTERFACE WSH_FREIGHT_CHARGES_INTERFACE

Table 1 – 1 Oracle Manufacturing Interfaces (Page 2 of 3)

Interface Name	Data Flow Direction	Table, View, or Process	Iface Man	TRM	Table, View, or Module Name
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
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 Service					
Service Request	Inbound	Processes	Yes	No	CS_ServiceRequest_PUB
Oracle Work in Process					
Moves	Inbound	Table	Yes	Yes	WIP_MOVE_TXN_INTERFACE
Resource	Inbound	Table	Yes	Yes	WIP_COST_TXN_INTERFACE
Job and Schedule	Inbound	Table	Yes	Yes	WIP_JOB_SCHEDULE_INTERFACE
Material	Inbound	Table	Yes	Yes	MTL_TRANSACTIONS_INTERFACE MTL_SERIAL_NUMBERS_INTERFACE MTL_TRANSACTION_LOTS_INTERFACE
WIP Scheduling	Outbound/ Inbound	Processes and Table	Yes	Yes	Stored Procedures LOAD_INTERFACE and LOAD_WIP and WIP_SCHEDULING_INTERFACE

Table 1 – 1 Oracle Manufacturing Interfaces (Page 3 of 3)

Inbound Open Interface Model

Oracle Manufacturing, Distribution, Sales and Service 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.

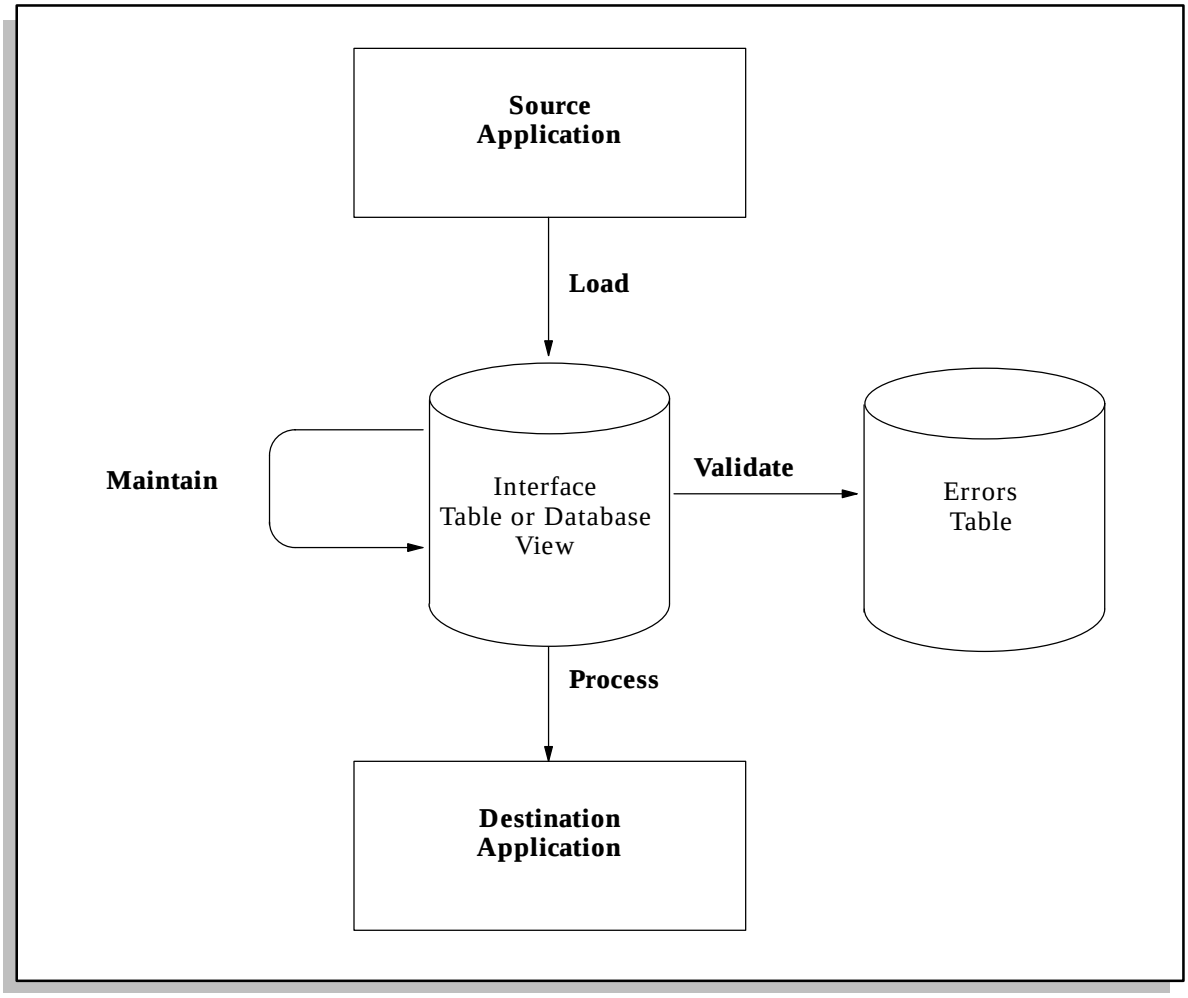
Discussion of Inbound Interfaces Only

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:

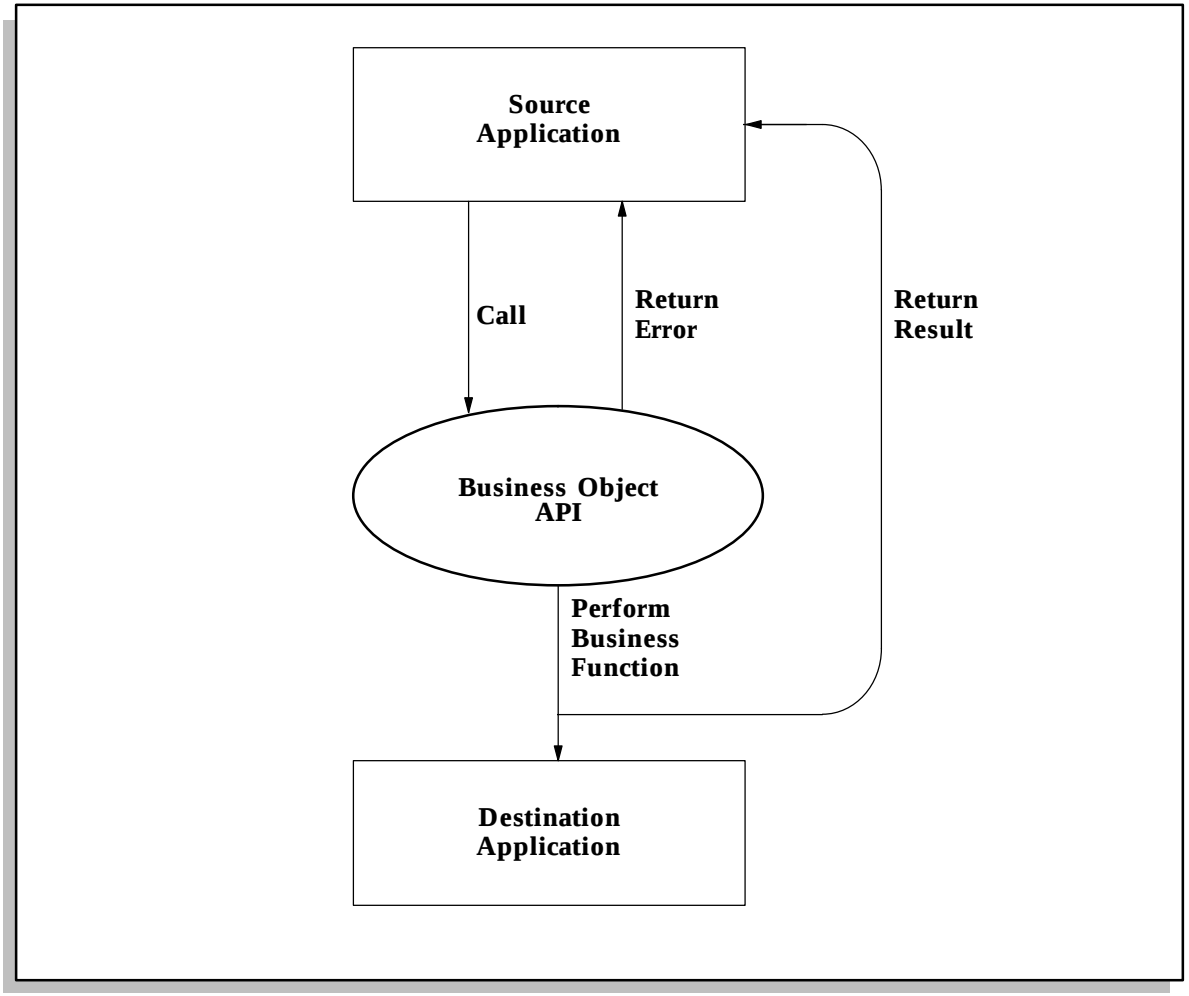
Figure 1 – 1


Source Application
Load
Maintain
Interface Table or Database View
Validate
Errors Table
Process
Destination Application

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



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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales or Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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 Entry/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, Distribution, Sales and Service source products available for selection and use by destination applications.

Oracle Manufacturing, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales and Service 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, Distribution, Sales or Service 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, Distribution, Sales or Service 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, Distribution, Sales or Service 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, Distribution, Sales or Service 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, Distribution, Sales or Service 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, Distribution, Sales or Service product is the source application, the destination application should provide the maintain function.

CHAPTER

2

Oracle Bills of Material Open Interfaces

This chapter contains information about the following Oracle Bills of Material open interfaces:

Open Bills of Material Interface

You can automatically import bills of material and product family members into Oracle Bills of Material or Oracle Engineering from any source using the Bill and Routing Interface. With this interface, you can easily convert manufacturing bills of material from legacy manufacturing systems, migrate marketing bills of material from custom sales order entry systems, import new engineering bills from Product Data Management (PDM) systems, or add members to product families. You can also update and delete existing bills and product families. Oracle Bills of Material and Oracle Engineering validates your data, ensuring that your imported bills of material or product families contain the same detail as those you enter manually in the Bills of Material, Engineering Bills of Material, or Product Families form.

The purpose of this essay is to explain how to use the Bill and Routing Interface so you can seamlessly integrate other applications with Oracle Bills of Material and Oracle Engineering.

See Also

Bills of Material, *Oracle Bills of Material User's Guide*

Engineering Prototype Environment, *Oracle Engineering User's Guide*

Functional Overview

Once you install Oracle Bills of Material and Oracle Engineering, you can use the Bill and Routing Interface program to create, update, or delete manufacturing and engineering bills of material and product families. Based on the attributes of the parent item, the program creates planning, model, option class, and standard bills of material. The Bill and Routing Interface validates your data the same way Oracle Bills of Material and Oracle Engineering verify bills of material and product families entered manually.

Attention: All parent items, product family items, and components must be defined in Oracle Inventory or Oracle Engineering before you can import a bill of material or product family.

Before you use the Bill and Routing Interface, you must write and run a custom program that extracts bill of material, product family, and

component details from your source system. This program must insert rows in the following tables for each extracted bill of material:

- BOM_BILL_OF_MTLS_INTERFACE
- BOM_INVENTORY_COMPS_INTERFACE
- BOM_REF_DESGS_INTERFACE
- BOM_SUB_COMPS_INTERFACE
- MTL_ITEM_REVISIONS_INTERFACE

When importing a product family, only the following tables are used:

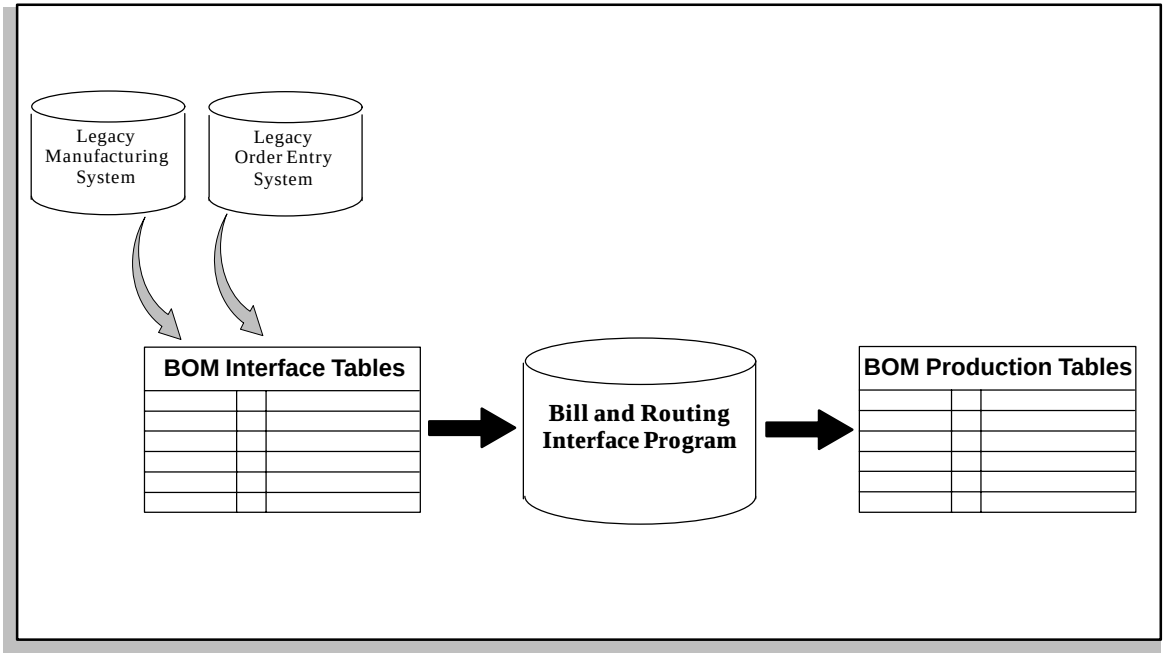
- BOM_BILL_OF_MTLS_INTERFACE
- BOM_INVENTORY_COMPS_INTERFACE

After you load the data in the interface tables, you can launch the Bill and Routing Interface program from the Import Bills and Routings form in Oracle Bills of Material or Oracle Engineering. The program assigns values, validates the data you include, and then creates, updates, or deletes the bills of material or product families.

You can optionally create an item revision when you import a bill of material, by inserting a value for a revision at the same time you create your bill of material data. If you enter a value in the REVISION column of the BOM_BILL_OF_MTLS_INTERFACE table, the Bill and Routing Interface program inserts a row into the MTL_ITEM_REVISIONS_INTERFACE table. To assign multiple item revisions, you can insert data directly into the MTL_ITEM_REVISIONS_INTERFACE table. If the validation is successful, the program then imports the data into the MTL_ITEM_REVISIONS table.

The following diagram displays the import process, from inserting data into the interface tables, through importing data into the Oracle Bills of Material and Oracle Engineering production tables.

Figure 2 – 1



See Also

Creating a Product Family, *Oracle Bills of Material User's Guide*

Defining a Bill of Material, *Oracle Bills of Material User's Guide*

Bills of Material, *Oracle Bills of Material User's Guide*

Engineering Prototype Environment, *Oracle Engineering User's Guide*

Creating a Bill or Product Family

To create a bill of material or a product family you must enter "Create" in the TRANSACTION_TYPE column. To create a bill of material or product family with components, you must populate the BOM_BILL_OF_MTLS_INTERFACE and BOM_INVENTORY_COMPS_INTERFACE tables. Using these two tables, you can create bill of material and product family header information and assign component details. You can also populate the BOM_REF_DESGS_INTERFACE, and BOM_SUB_COMPS_INTERFACE tables to assign reference designators and substitute components to your bills of material.

Updating a Bill or Product Family

To update a bill or family, you must enter ”Update” in the TRANSACTION_TYPE column.

To update a column to a NULL value, you must enter the following values:

Column Data Type	Required value to create NULL column
For Char columns	chr(12)
For Date columns	TO_DATE('1','j')
For Number columns	9.99E125

Table 2 – 1 Bill of Material Interface, Null Values (Page 1 of 1)

Deleting a Bill or Product Family

To delete a bill or product family, you must enter ”Delete” in the TRANSACTION_TYPE column. You must enter delete group names for bills, components and families.

Validation

After populating the interface tables, you can run the Bill and Routing Interface program. The column, PROCESS_FLAG, indicates the current state of processing for a row in the interface table. Possible values for the column include:

1 – Pending

2 – Assign Succeeded

3 – Assign / Validation Failed

4 – Import Failed

7 – Import Succeeded

All the inserted records must have the PROCESS_FLAG set to **1** (Pending). The program assigns and validates all rows with a status of **1** (Pending) and then imports them into the production tables. If the assign or validate procedure fails for a row, the program sets the PROCESS_FLAG to **3** (Assign / Validation Failed) for that row. The successful rows continue through the process of importing into the production tables. If a row fails on import, the program assigns a value

of **4** (Import Failed) to the PROCESS_FLAG. Successfully imported rows have a value of **7** (Import Succeeded) for the PROCESS_FLAG.

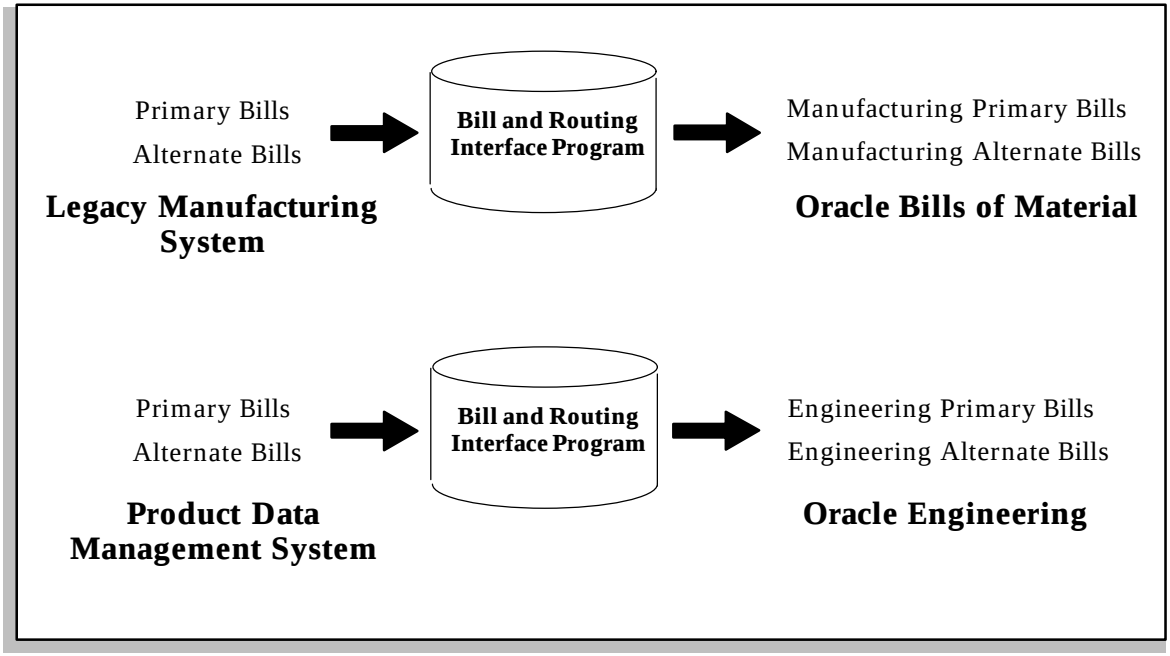
When you submit the Bill and Routing Interface program, Oracle Bills of Material or Oracle Engineering automatically updates the TRANSACTION_ID and REQUEST_ID columns in each of the interface tables. The column TRANSACTION_ID stores a unique id for each row in the interface table and the REQUEST_ID column stores the concurrent request id number.

Although you can import bills and routings simultaneously, all routing operations must exist before you can assign a component to an operation. If a routing does not exist, you cannot assign an operation sequence to a component on a bill of material.

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

The Bill and Routing Interface program lets you import manufacturing and engineering bills of material. Specify a value in the ASSEMBLY_TYPE column to determine the type of bill you want to import. If you import a manufacturing bill of material, the bill is visible to both Oracle Bills of Material and Oracle Engineering. However, if you import an engineering bill of material, it is only accessible through the Oracle Engineering product. The following diagram illustrates the data the Bill and Routing interface program processes and imports into Oracle Bills of Material and Oracle Engineering.

Figure 2 – 2



See Also

Bills of Material, *Oracle Bills of Material User's Guide*

Defining a Bill of Material, *Oracle Bills of Material User's Guide*

Routings, *Oracle Bills of Material User's Guide*

Oracle Bills of Material Technical Reference Manual

Setting Up for Bills of Material Import

When you create, update, or delete bills of material or product families, there are no additional setup steps in Oracle Bills of Material or Oracle Engineering beyond those already required to manually define manufacturing and engineering bills of material. You must define all items before they can be assigned to a manufacturing or engineering bill of material or product family. Since you launch and manage the Bill and Routing Interface program through the concurrent manager,

you must insure that the concurrent manager is running before you can import any bills.

See Also

Bills of Material, *Oracle Bills of Material User's Guide*

Setting Up Oracle Bills of Material, *Oracle Bills of Material User's Guide*

Setting Up Oracle Engineering, *Oracle Engineering User's Guide*

Bill and Routing Interface Runtime Options

When you run the Bill and Routing Interface program, you must specify a number of runtime options. These options include:

All Organizations

Yes	Run the interface for all organization codes in the bill of material and routing interface tables.
No	Run the interface program only for the organization you are currently in. The interface program only processes bill of material and routing interface records in your current organization.

Attention: When you specify **No** for this option, the program ignores rows in the interface tables that do not have an organization code or organization id assigned.

Import Routings

Yes	Import records from the routing interface tables for your current organization or all organizations.
No	Do not import records from the routing interface tables.

Import Bills of Material

Yes	Import records from the bills of material interface tables for your current organization or all organizations.
No	Do not import records from the bills of material interface tables.

Delete Processed Rows

Yes Delete successfully processed rows from the bill of material and routing interface tables.

No Leave all records in the bill of material and routing interface tables for successfully processed rows.

Inserting into BOM_BILL_OF_MTLS_INTERFACE Table

You must load the bill of material or product family information you want to import from your source system into the BOM_BILL_OF_MTLS_INTERFACE and BOM_INVENTORY_COMPS_INTERFACE tables. Oracle Bills of Material derives any additional data, validates your data, and then creates, updates, or deletes bills of material and product families.

BOM_BILLS_OF_MTLS_INTERFACE Table Description

The following graphic describes the columns in the BOM_BILL_OF_MTLS_INTERFACE table that you use to insert data:

BOM_BILL_OF_MTLS_ INTERFACE Column Name	Type	Derived or Defaulted Value
ASSEMBLY_ITEM_ID	Number	From ITEM_NUMBER
ORGANIZATION_ID	Number	From ORGANIZATION_CODE
ALTERNATE_BOM_DESIGNATOR	Varchar (10)	
LAST_UPDATE_DATE	Date	System Date
LAST_UPDATED_BY	Number	Userid
CREATION_DATE	Date	System Date
CREATED_BY	Number	Userid
PROGRAM_APPLICATION_ID	Number	
PROGRAM_ID	Number	
PROGRAM_UPDATE_DATE	Date	
COMMON_ASSEMBLY_ITEM_ID	Number	From COMMON_ITEM_NUMBER

Table 2 – 2 BOM Bill of Material Interface (Page 1 of 3)

BOM_BILL_OF_MTLS_ INTERFACE Column Name	Type	Derived or Defaulted Value
SPECIFIC_ASSEMBLY_COMMENT	Varchar (240)	
TRANSACTION_TYPE	Varchar2 (10)	
ATTRIBUTE_CATEGORY	Varchar (30)	
ATTRIBUTE1	Varchar (150)	
ATTRIBUTE2	Varchar (150)	
ATTRIBUTE3	Varchar (150)	
ATTRIBUTE4	Varchar (150)	
ATTRIBUTE5	Varchar (150)	
ATTRIBUTE6	Varchar (150)	
ATTRIBUTE7	Varchar (150)	
ATTRIBUTE8	Varchar (150)	
ATTRIBUTE9	Varchar (150)	
ATTRIBUTE10	Varchar (150)	
ATTRIBUTE11	Varchar (150)	
ATTRIBUTE12	Varchar (150)	
ATTRIBUTE13	Varchar (150)	
ATTRIBUTE14	Varchar (150)	
ATTRIBUTE15	Varchar (150)	
ASSEMBLY_TYPE	Number	1
COMMON_BILL_SEQUENCE_ID	Number	BILL_SEQUENCE_ID
BILL_SEQUENCE_ID	Number	For Creates: Defaulted from sequence BOM_INVENTORY_COMPONENTS_S For Updates and Deletes: Derived from ASSEMBLY_ITEM_ID, ORGANIZATION_ID, ALTERNATE_BOM_DESIGNATOR
REVISION	Varchar (3)	
COMMON_ORGANIZATION_ID	Number	From COMMON_ORG_CODE
PROCESS_FLAG	Number	

Table 2 – 2 BOM Bill of Material Interface (Page 2 of 3)

BOM_BILL_OF_MTLS_ INTERFACE Column Name	Type	Derived or Defaulted Value
ORGANIZATION_CODE	Varchar (3)	
COMMON_ORG_CODE	Varchar (3)	
ITEM_NUMBER	Varchar (81)	
COMMON_ITEM_NUMBER	Varchar (81)	
REQUEST_ID	Number	From FND_CONCURRENT_REQUESTS

Table 2 – 2 BOM Bill of Material Interface (Page 3 of 3)

Required Data

Creating a Bill

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

- PROCESS_FLAG
- ORGANIZATION_ID
- ASSEMBLY_ITEM_ID
- ASSEMBLY_TYPE
- TRANSACTION_TYPE (Create)

If you are creating an alternate bill of material you must also enter a value in the ALTERNATE_BOM_DESIGNATOR column.

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

When you insert new rows into the BOM_BILL_OF_MTLS_INTERFACE table, you should set the PROCESS_FLAG to **1** (Pending).

Creating a Product Family

When creating a product family, you must enter values in the following required columns:

- ASSEMBLY_ITEM_ID or ITEM_NUMBER
- ORGANIZATION_ID or ORGANIZATION_CODE
- TRANSACTION_TYPE (Create)
- PROCESS_FLAG (1)

Updating a Bill

To identify the bill record you are trying to update, you must provide one of the following:

- BILL_SEQUENCE_ID or
- ASSEMBLY_ITEM_ID or ITEM_NUMBER,

  ALTERNATE_BOM_DESIGNATOR, and

  ORGANIZATION_ID or ORGANIZATION_CODE

If you only enter COMMON_ASSEMBLY_ITEM_ID and not the organization, the program will assume the common organization is the organization that the bill belongs to. If you want to null out the common bill, enter 9.99E125 in the COMMON_BILL_SEQUENCE_ID.

You must always enter values for the following required columns when you update rows in the BOM_BILL_OF_MTLS_INTERFACE table

- TRANSACTION_TYPE (Update)
- PROCESS_FLAG (1)

Updating a Product Family

When updating a product family, you must enter values in the following required columns:

- ASSEMBLY_ITEM_ID or ITEM_NUMBER
- ORGANIZATION_ID or ORGANIZATION_CODE
- TRANSACTION_TYPE (Update)
- PROCESS_FLAG (1)

Deleting a Bill

To identify the bill record you are trying to delete, you must provide one of the following:

- BILL_SEQUENCE_ID or
- ALTERNATE_BOM_DESIGNATOR,

  ASSEMBLY_ITEM_ID or ITEM_NUMBER, and

  ORGANIZATION_ID or ORGANIZATION_CODE

You must always enter values for the following required column when you delete rows in the BOM_BILL_OF_MTLS_INTERFACE table

- TRANSACTION_TYPE (Delete)
- PROCESS_FLAG (1)

You must insert a record in the BOM_INTERFACE_DELETE_GROUPS table with the following values:

- ENTITY_NAME (BOM_BILL_OF_MTLS_INTERFACE)
- DELETE_GROUP_NAME (A new name or name of an existing Delete Group for bills)
- DESCRIPTION (Required if using a new delete group)

The import program will then create a delete group for the bill or add the bill to an existing delete group.

Deleting a Product Family

To identify the product family record you are trying to delete, you must provide one of the following:

- BILL_SEQUENCE_ID or
- ASSEMBLY_ITEM_ID or ITEM_NUMBER, and

  ORGANIZATION_ID or ORGANIZATION_CODE

You must always enter values for the following required column when you delete rows in the BOM_BILL_OF_MTLS_INTERFACE table

- TRANSACTION_TYPE (Delete)
- PROCESS_FLAG (1)

You must insert a record in the BOM_INTERFACE_DELETE_GROUPS table with the following values:

- ENTITY_NAME (BOM_BILL_OF_MTLS_INTERFACE)

- DELETE_GROUP_NAME (A new name or name of an existing Delete Group for bills)
- DESCRIPTION (Required if using a new delete group)

The import program will then create a delete group for the bill or add the bill to an existing delete group.

Derived Data

Creating a Bill

The Bill and Routing Interface program derives or defaults most of the data required to create a manufacturing or an engineering bill of material. The Bill and Routing Interface program derives or defaults the columns using the same logic as the Bills of Material window. When you populate a column in the interface table, the program imports the row with the data you include and does not default a value. However, if you do not enter data in a derived or defaulted column, the program automatically imports the row with the derived or defaulted value. For example, if you do not specify a value for the CREATION_DATE column, the Bill and Routing Interface program automatically defaults the current date.

The BOM_BILL_OF_MTLS_INTERFACE table contains user–friendly columns that let you easily enter data in the interface table. The Bill and Routing Interface program uses the values you enter for these columns to derive unique identifiers. For example, when you import a bill of material you can include a value for ORGANIZATION_CODE or ORGANIZATION_ID. If you enter a value for ORGANIZATION_CODE, the Bill and Routing Interface program derives the value for ORGANIZATION_ID. The following table lists these columns and the corresponding column that stores the derived value:

BOM_BILL_OF_MTLS_INTERFACE User–Friendly Column Name	Derived Column Name
ORGANIZATION_CODE	ORGANIZATION_ID
ITEM_NUMBER	ASSEMBLY_ITEM_ID

Table 2 – 3 BOM Bill of Material Interface, Derived Columns (Page 1 of 2)

BOM_BILL_OF_MTLS_INTERFACE User–Friendly Column Name	Derived Column Name
COMMON_ORG_CODE	COMMON_ORGANIZATION_ID
COMMON_ITEM_NUMBER	COMMON_ASSEMBLY_ITEM_ID

Table 2 – 3 BOM Bill of Material Interface, Derived Columns (Page 2 of 2)

Attention: If you enter a value for the ITEM_NUMBER or COMMON_ITEM_NUMBER column, you must insert the system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for an item, it searches for this separator to indicate the end of one segment value and the start of the next segment value.

The Bill and Routing Interface program derives values for some columns in BOM_BILL_OF_MTLS_INTERFACE based on values entered for other columns. For example, the program derives the COMMON_BILL_SEQUENCE_ID column from the following columns:

- COMMON_ITEM_ID
- COMMON_ORGANIZATION_ID
- ALTERNATE_BOM_DESIGNATOR

You can specify in the ASSEMBLY_TYPE column whether the bill of material is a manufacturing bill or an engineering bill. If you do not include a value for this column, Oracle Bills of Material defaults a value of **1** (manufacturing), and creates a manufacturing bill of material. To create an engineering bill, you must enter a value of **2** (engineering) for the ASSEMBLY_TYPE column. When you import a bill of material, the BOM Item Type attribute of the parent item determines the type of bill created, including planning, model, option class, and standard bills. For example, if you import a model item and then import its bill, the Bill and Routing Interface program creates the bill as a model bill of material.

Creating a Product Family

The Bill and Routing Interface program derives or defaults most of the data required to create a product family. The Bill and Routing Interface program derives or defaults the columns using the same logic as the Product Family window.

The BOM_BILL_OF_MTLS_INTERFACE table contains user–friendly columns that let you easily enter data in the interface table. The Bill and Routing Interface program uses the values you enter for these columns to derive unique identifiers. For example, when you import a product family you can include a value for ORGANIZATION_CODE or ORGANIZATION_ID. If you enter a value for ORGANIZATION_CODE, the Bill and Routing Interface program derives the value for ORGANIZATION_ID. The following table lists these columns and the corresponding column that stores the derived value:

BOM_BILL_OF_MTLS_INTERFACE User–Friendly Column Name	Derived Column Name
ORGANIZATION_CODE	ORGANIZATION_ID
ITEM_NUMBER	ASSEMBLY_ITEM_ID

Table 2 – 4 BOM Bill of Material Interface, Product Family Derived Columns (Page 1 of 1)

Updating a Bill

The Bill and Routing Interface program derives the same data when updating a bill as it does when creating a bill.

Updating a Product Family

The Bill and Routing Interface program derives the same data when updating a product family as it does when creating a product family.

Deleting a Bill

The Bill and Routing Interface program derives the same data when deleting a bill as it does when creating a bill.

Deleting a Product Family

The Bill and Routing Interface program derives the same data when deleting a product family as it does when creating a product family.

See Also

Define Key Flexfield Segments, *Oracle Applications Flexfields Manual*

Optional Data

Creating a Bill

You can optionally assign a value to the other columns in the interface table or let the Bill and Routing Interface program default a value.

You can insert a value in the REVISION column to update the parent item revision. The Bill and Routing Interface program inserts a row into the MTL_ITEM_REVISIONS_INTERFACE table, validates the data and then imports the record into the MTL_ITEM_REVISIONS table. To insert multiple item revisions for a bill, you can insert the data directly into the MTL_ITEM_REVISIONS_INTERFACE table.

You can also populate the descriptive flexfield segment columns in the interface table to import values for the bill of material descriptive flexfield. You define the structure for the descriptive flexfield in the column ATTRIBUTE_CATEGORY and you can populate each segment value in the columns ATTRIBUTE1 through ATTRIBUTE15. After importing a bill, you can view your descriptive flexfield values in the Bill of Material block of the Bills of Material form and the Engineering Bills of Material form.

Creating a Product Family

The Bill and Routing Interface program contains the same optional data when creating a product family as it does when creating a bill.

Attention: The optional columns for a product family are a subset of the optional columns for a bill of material.

Updating a Bill

The following columns are updatable:

- SPECIFIC_ASSEMBLY_COMMENT (Nullable)
- COMMON_BILL_SEQUENCE_ID (Nullable)
- LAST_UPDATE_DATE
- LAST_UPDATED_BY
- LAST_UPDATE_LOGIN

- ATTRIBUTE_CATEGORY
- ATTRIBUTE1 through ATTRIBUTE15
- REQUEST_ID
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE

To null out the common bill, enter 9.99E125 in the COMMON_BILL_SEQUENCE_ID column.

Updating a Product Family

The following columns are updatable:

- LAST_UPDATE_DATE
- LAST_UPDATED_BY
- LAST_UPDATE_LOGIN
- ATTRIBUTE_CATEGORY
- ATTRIBUTE1 through ATTRIBUTE15
- REQUEST_ID
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE

Deleting a Bill

Once the bill is identified, the entire record will be placed in the specified delete group. The Interface program will not automatically delete the bill.

Deleting a Product Family

Once the bill is identified, the entire record will be placed in the specified delete group. The Interface program will not automatically delete the product family.

See Also

Oracle Bills of Material Technical Reference Manual

Inserting into BOM_INVENTORY_COMPS_INTERFACE Table

You must load the component or product family member details for each bill of material in the BOM_INVENTORY_COMPS_INTERFACE table. Oracle Bills of Material and Oracle Engineering use this component information, along with the data in the BOM_BILL_OF_MTLS_INTERFACE table, to create new bills of material and update or delete existing components.

BOM_INVENTORY_COMPS_INTERFACE Table Description

The following graphic describes the columns in the BOM_INVENTORY_COMPS_INTERFACE table that you use to insert data:

BOM_INVENTORY_COMPS_ INTERFACE Column Name	Type	Derived or Defaulted Value
OPERATION_SEQ_NUM	Number	
COMPONENT_ITEM_ID	Number	FromCOMPONENT_ITEM_NUMBER
LAST_UPDATE_DATE	Date	System Date
LAST_UPDATED_BY	Number	Userid
LAST_UPDATE_LOGIN	Varchar (10)	
CREATION_DATE	Date	System Date
CREATED_BY	Number	Userid
PROGRAM_APPLICATION_ID	Number	
PROGRAM_ID	Number	
PROGRAM_UPDATE_DATE	Date	
ITEM_NUM	Number	1
COMPONENT_QUANTITY	Number	1
COMPONENT_YIELD_FACTOR	Number	1
COMPONENT_REMARKS	Varchar (240)	
EFFECTIVITY_DATE	Date	System Date
DISABLE_DATE	Date	

Table 2 – 5 BOM Inventory Components Interface (Page 1 of 4)

BOM_INVENTORY_COMPS_ INTERFACE Column Name	Type	Derived or Defaulted Value
IMPLEMENTATION_DATE	Date	
NEW_OPERATION_SEQ_NUM	Number	(Only used for updates)
NEW_EFFECTIVITY_DATE	Date	(Only used for updates)
ASSEMBLY_TYPE	Number	(Used by the program)
TRANSACTION_TYPE	Varchar2(10)	
INTERFACE_ENTITY_TYPE	Varchar2(4)	
ATTRIBUTE CATEGORY	Varchar (30)	
ATTRIBUTE1	Varchar (150)	
ATTRIBUTE2	Varchar (150)	
ATTRIBUTE3	Varchar (150)	
ATTRIBUTE4	Varchar (150)	
ATTRIBUTE5	Varchar (150)	
ATTRIBUTE6	Varchar (150)	
ATTRIBUTE7	Varchar (150)	
ATTRIBUTE8	Varchar (150)	
ATTRIBUTE9	Varchar (150)	
ATTRIBUTE10	Varchar (150)	
ATTRIBUTE11	Varchar (150)	
ATTRIBUTE12	Varchar (150)	
ATTRIBUTE13	Varchar (150)	
ATTRIBUTE14	Varchar (150)	
ATTRIBUTE15	Varchar (150)	
PLANNING_FACTOR	Number	100
QUANTITY_RELATED	Number	2
SO_BASIS	Number	2
OPTIONAL	Number	2

Table 2 – 5 BOM Inventory Components Interface (Page 2 of 4)

BOM_INVENTORY_COMPS_ INTERFACE Column Name	Type	Derived or Defaulted Value
MUTUALLY_EXCLUSIVE_OPTIONS	Number	2
INCLUDE_IN_COST_ROLLUP	Number	1
CHECK_ATP	Number	2
REQUIRED_TO_SHIP	Number	2
REQUIRED_FOR_REVENUE	Number	2
INCLUDE_ON_SHIP_DOCS	Number	2
LOW_QUANTITY	Number	
HIGH_QUANTITY	Number	
COMPONENT_SEQUENCE_ID	Number	For Creates: Defaults from sequence BOM_INVENTORY_COMPONENTS_S For Updates and Deletes: Derived from COMPONENT_ITEM_ID, EFFECTIVITY_DATE,BILL_SEQUENCE_ID, and OPERATION_SEQ_NUM
BILL_SEQUENCE_ID	Number	From BOM_BILL_OF_MTLS_INTERFACE or BOM_BILL_OF_MATERIALS
WIP_SUPPLY_TYPE	Number	1
PICK_COMPONENTS	Varchar (10)	
BOM_ITEM_TYPE	Number	
SUPPLY_SUBINVENTORY	Varchar (10)	
SUPPLY_LOCATOR_ID	Number	From LOCATION_NAME
OPERATION_LEAD_TIME_PERCENT	Number	
ASSEMBLY_ITEM_ID	Number	From ASSEMBLY_ITEM_NUMBER
ALTERNATE_BOM_DESIGNATOR	Varchar (10)	
ORGANIZATION_ID	Number	From ORGANIZATION_CODE
ORGANIZATION_CODE	Varchar (3)	
COMPONENT_ITEM_NUMBER	Varchar (81)	
ASSEMBLY_ITEM_NUMBER	Varchar (81)	

Table 2 – 5 BOM Inventory Components Interface (Page 3 of 4)

BOM_INVENTORY_COMPS_ INTERFACE Column Name	Type	Derived or Defaulted Value
LOCATION_NAME	Varchar (81)	
REFERENCE_DESIGNATOR	Varchar (15)	
SUBSTITUTE_COMP_ID	Number	FromSUBSTITUTE_COMP_NUMBER
SUBSTITUTE_COMP_NUMBER	Varchar (81)	
PROCESS_FLAG	Number	
REQUEST_ID	Number	From FND_CONCURRENT_REQUESTS

Table 2 – 5 BOM Inventory Components Interface (Page 4 of 4)

Attention: You cannot dynamically create locations for your subinventories using the interface tables. If you specify a value for the LOCATION_ID column, the location must already exist in Oracle Inventory.

Required Data

Creating a Component

Each imported record must have a value for the following columns:

- COMPONENT_ITEM_ID
- COMPONENT_SEQUENCE_ID
- OPERATION_SEQ_NUM
- EFFECTIVITY_DATE
- BILL_SEQUENCE_ID
- TRANSACTION_TYPE (Create)
- PROCESS_FLAG (1)

You also must specify a value in the ALTERNATE_BOM_DESIGNATOR column if you assign components to an alternate bill of material and have not entered a value for the BILL_SEQUENCE_ID column.

When you insert rows into BOM_INVENTORY_COMPS_INTERFACE, you must set the PROCESS_FLAG to **1** (Pending) for the Bill and Routing Interface program to process the record.

Creating a Product Family Member

Each imported record must have a value for the following columns:

- COMPONENT_ITEM_ID
- BILL_SEQUENCE_ID
- TRANSACTION_TYPE (Create)
- PROCESS_FLAG (1)

Updating a Component

To identify the component record you are trying to update, you must provide one of the following:

- COMPONENT_SEQUENCE_ID or
- BILL_SEQUENCE_ID,

  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,

  OPERATION_SEQ_NUM, and

  EFFECTIVITY_DATE
- ASSEMBLY_ITEM_ID or ASSEMBLY_ITEM_NUMBER,

  ALTERNATE_BOM_DESIGNATOR,

  ORGANIZATION_ID or ORGANIZATION_CODE,

  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,

  OPERATION_SEQ_NUM, and

  EFFECTIVITY_DATE

Attention: If the component is unimplemented, you cannot update the record.

If you would like to update the OPERATION_SEQ_NUM or EFFECTIVITY_DATE, then you must fill in NEW_OPERATION_SEQ_NUM or NEW_EFFECTIVITY_DATE.

You must always enter values for the following required columns when you update rows in the BOM_INVENTORY_COMPS_INTERFACE table

- TRANSACTION_TYPE (Update)

- PROCESS_FLAG (1)

Updating a Product Family Member

To identify the product family member you are trying to update, you must provide one of the following:

- COMPONENT_SEQUENCE_ID or
- BILL_SEQUENCE_ID,
  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,
  and EFFECTIVITY_DATE or
- ASSEMBLY_ITEM_ID or ASSEMBLY_ITEM_NUMBER,
  ORGANIZATION_ID or ORGANIZATION_CODE,
  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,
  and EFFECTIVITY_DATE

Attention: If the component is unimplemented, you cannot update the record.

You must always enter values for the following required column when you update rows in the BOM_INVENTORY_COMPS_INTERFACE table

- TRANSACTION_TYPE (Update)
- PROCESS_FLAG (1)

Deleting a Component

To identify the component record you are trying to delete, you must provide one of the following:

- COMPONENT_SEQUENCE_ID or
- BILL_SEQUENCE_ID,
  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,
  OPERATION_SEQ_NUM, and
  EFFECTIVITY_DATE or
- ASSEMBLY_ITEM_ID or ASSEMBLY_ITEM_NUMBER,
  ALTERNATE_BOM_DESIGNATOR,
  ORGANIZATION_ID or ORGANIZATION_CODE,
  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,

OPERATION_SEQ_NUM, and

EFFECTIVITY_DATE

Attention: If the component is unimplemented, you cannot delete the record.

You must always enter values for the following required column when you delete rows in the BOM_INVENTORY_COMPS_INTERFACE table

- TRANSACTION_TYPE (Delete)
- PROCESS_FLAG (1)

You must insert a record in the BOM_INTERFACE_DELETE_GROUPS table with the following values:

- ENTITY_NAME (BOM_INVENTORY_COMPS_INTERFACE)
- DELETE_GROUP_NAME (A new name or name of an existing Delete Group for components)
- DESCRIPTION (Required if using a new delete group)

The import program will then create a delete group for the component or add the component to an existing delete group.

Deleting a Product Family Member

To identify the product family member you are trying to delete, you must provide one of the following:

- COMPONENT_SEQUENCE_ID or
- BILL_SEQUENCE_ID,

  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,

  and EFFECTIVITY_DATE or
- ASSEMBLY_ITEM_ID or ASSEMBLY_ITEM_NUMBER,

  ORGANIZATION_ID or ORGANIZATION_CODE,

  COMPONENT_ITEM_ID or COMPONENT_ITEM_NUMBER,

  and EFFECTIVITY_DATE

Attention: If the component is unimplemented, you cannot delete the record.

You must always enter values for the following required column when you delete rows in the BOM_INVENTORY_COMPS_INTERFACE table

- TRANSACTION_TYPE (Delete)

- PROCESS_FLAG (1)

Derived Data

Creating a Componet

The Bill and Routing Interface program derives or defaults most of the data required to assign components to a bill of material. You can optionally include a value for any derived or defaulted column, as well as data for any of the other columns in the interface table.

The Interface program uses the same logic to derive or default column values in the BOM_INVENTORY_COMPS_INTERFACE table as in the BOM_BILL_OF_MTLS_INTERFACE table; when you populate a column in the interface table, the Bill and Routing Interface program imports the row with the data you include. However, if you do not enter data in a derived or defaulted column, the program automatically imports the row with the derived or defaulted value.

The BOM_INVENTORY_COMPS_INTERFACE table contains columns that allow you to enter the user–friendly value in the interface table and the program derives the associated system's unique identifier. The following table lists these columns and the corresponding column that stores the derived value:

BOM_INVENTORY_COMPS_INTERFACE User–Friendly Column Name	**Derived Column Name**
ORGANIZATION_CODE	ORGANIZATION_ID
ASSEMBLY_ITEM_NUMBER	ASSEMBLY_ITEM_ID
COMPONENT_ITEM_NUMBER	COMPONENT_ITEM_ID
SUBSTITUTE_COMP_NUMBER	SUBSTITUTE_COMP_ID
LOCATION_NAME	SUPPLY_LOCATOR_ID

Table 2 – 6 BOM Inventory Components Interface, Derived Columns (Page 1 of 1)

Attention: If you enter a value for the COMPONENT_ITEM_NUMBER, ASSEMBLY_ITEM_NUMBER, and SUBSTITUTE_COMP_NUMBER columns, you must insert the

system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for the item, it searches for this separator to indicate the end of one segment value and the start of the next segment value.

When you insert data into the columns LOW_QUANTITY and HIGH_QUANTITY, the program validates that the value for LOW_QUANTITY is less than the value for HIGH_QUANTITY. If you insert a value for only one of these columns, the Bill and Routing Interface program inserts the value into both columns when importing the record.

For a detailed description of the derived or defaulted columns, please refer to the Oracle Bills of Material Technical Reference Manual.

Creating a Product Family Member

The BOM_INVENTORY_COMPS_INTERFACE table contains columns that allow you to enter the user–friendly value in the interface table and the program derives the associated system's unique identifier. The following table lists these columns and the corresponding column that stores the derived value:

BOM_INVENTORY_COMPS_INTERFACE User–Friendly Column Name	**Derived Column Name**
ORGANIZATION_CODE	ORGANIZATION_ID
ASSEMBLY_ITEM_NUMBER	ASSEMBLY_ITEM_ID
COMPONENT_ITEM_NUMBER	COMPONENT_ITEM_ID
ASSEMBLY_ITEM_ID ORGANIZATION_ID ALTERNATE_BOM_DESIGNATOR	BILL_SEQUENCE_ID

Table 2 – 7 BOM Inventory Components Interface Table, Derived Columns for Product Family Members (Page 1 of 1)

Attention: If you enter a value for the COMPONENT_ITEM_NUMBER and ASSEMBLY_ITEM_NUMBER columns, you must insert the system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for the item, it searches for this separator to indicate the end of one segment value and the start of the next segment value.

Updating a Component

The Bill and Routing Interface program derives the same data when updating a component as it does when creating a component.

Although the BILL_SEQUENCE_ID and COMPONENT_SEQUENCE_ID are required columns, the Bill and Routing Interface program derives values for the columns when you do not include a value. When updating a component, the program derives a value for the BILL_SEQUENCE_ID from the following columns:

- ASSEMBLY_ITEM_ID
- ORGANIZATION_ID
- ALTERNATE_BOM_DESIGNATOR

COMPONENT_SEQUENCE_ID is derived from the following columns:

- BILL_SEQUENCE_ID
- COMPONENT_ITEM_ID
- OPERATION_SEQ_NUM
- EFFECTIVITY_DATE

Updating a Product Family Member

The Bill and Routing Interface program derives the same data when updating a product family member as it does when creating a product family member. When updating, the Bill and Routing Interface derives a value for the COMPONENT_SEQUENCE_ID from the following columns:

- BILL_SEQUENCE_ID
- COMPONENT_ITEM_ID
- OPERATION_SEQ_NUM
- EFFECTIVITY_DATE

Attention: If you enter a value for the COMPONENT_ITEM_NUMBER and ASSEMBLY_ITEM_NUMBER columns, you must insert the system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for the item, it searches for this separator to indicate the end of one segment value and the start of the next segment value.

Deleting a Component

The Bill and Routing Interface program derives the same data when deleting a component as it does when creating a component.

Although the BILL_SEQUENCE_ID and COMPONENT_SEQUENCE_ID are required columns, the Bill and Routing Interface program derives values for the columns when you do not include a value. When deleting a component, the program derives a value for the BILL_SEQUENCE_ID from the following columns:

- ASSEMBLY_ITEM_ID
- ORGANIZATION_ID
- ALTERNATE_BOM_DESIGNATOR

COMPONENT_SEQUENCE_ID is derived from the following columns:

- BILL_SEQUENCE_ID
- COMPONENT_ITEM_ID
- OPERATION_SEQ_NUM
- EFFECTIVITY_DATE

Deleting a Product Family Member

The Bill and Routing Interface program derives the same data when deleting a product family member as it does when updating a product family member.

See Also

Define Key Flexfield Segments, *Oracle Applications Flexfields Manual*

Oracle Bills of Material Technical Reference Manual

Oracle Engineering Technical Reference Manual

Optional Data

Creating a Component

You can import substitute component information using the BOM_INVENTORY_COMPS_INTERFACE table if each component has one substitute component and the substitute quantity is equal to the component quantity. To import multiple substitute components for a

component, you must insert rows in the BOM_SUB_COMPS_INTERFACE table.

You can use the REFERENCE_DESIGNATOR column in the BOM_INVENTORY_COMPS_INTERFACE table to assign one reference designator, regardless of the component quantity. If you want to import multiple reference designators for each component, you need to insert data in the BOM_REF_DESGS_INTERFACE table. See the next section "Importing Additional Bill Information" for more details.

When you implement a descriptive flexfield in the Component block of the Bills of Material form or the Engineering Bills of Material form, you can import the descriptive flexfield values using the interface table. You define the structure for the descriptive flexfield in the column ATTRIBUTE_CATEGORY and you can populate each segment value in the columns ATTRIBUTE1 through ATTRIBUTE15. After importing a bill, you can view your descriptive flexfield values in the Components block of the Define Bill of Material form.

Creating a Product Family Member

The Bill and Routing Interface program contains the same optional data for creating a product family member as it does for creating a component.

Updating a Component

The following columns are updatable:

- ITEM_NUM
- OPERATION_SEQ_NUM
- COMPONENT_QUANTITY
- EFFECTIVITY_DATE
- DISABLE_DATE
- PLANNING_FACTOR
- COMPONENT_YIELD_FACTOR
- INCLUDE_IN_COST_ROLLUP
- WIP_SUPPLY_TYPE
- SUPPLY_SUBINVENTORY
- SUPPLY_LOCATOR_ID
- CHECK_ATP

- OPTIONAL
- MUTUALLY_EXCLUSIVE_OPTIONS
- LOW_QUANTITY
- HIGH_QUANTITY
- SO_BASIS
- INCLUDE_ON_SHIP_DOCS
- REQUIRED_TO_SHIP
- REQUIRED_FOR_REVENUE
- COMPONENT_REMARKS
- QUANTITY_RELATED
- LAST_UPDATE_DATE
- LAST_UPDATE_BY
- LAST_UPDATE_LOGIN
- ATTRIBUTE_CATEGORY
- ATTRIBUTE1 through ATTRIBUTE15
- REQUEST_ID
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE

Updating a Product Family Member

The following columns are updatable:

- PLANNING_FACTOR
- DISABLE_DATE
- EFFECTIVITY_DATE
- COMPONENT_REMARKS
- LAST_UPDATE_DATE
- LAST_UPDATE_BY
- LAST_UPDATE_LOGIN
- ATTRIBUTE_CATEGORY
- ATTRIBUTE1 through ATTRIBUTE15

- REQUEST_ID
- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE

Deleting a Component

Once the component is identified the entire record will be placed in a specified delete group. The program will not automatically delete the component.

Deleting a Product Family Member

Once the product family member is identified, the program will automatically delete the product family member.

See Also

Oracle Bills of Material Technical Reference Manual

Importing Additional Bill Information

When you create bills of material and assign components using the Bill and Routing Interface program, you can also import additional bill of material information using three different interface tables. And, you can assign component reference designators using the BOM_REF_DESGS_INTERFACE table and substitute components using the BOM_SUB_COMPS_INTERFACE table. You can also insert multiple item revisions using the MTL_ITEM_REVISIONS_INTERFACE table. Oracle Bills of Material and Oracle Engineering validates this information the same way the forms validate bill information.

Inserting Bill of Material Details

You can insert values into the BOM_REF_DESGS_INTERFACE, BOM_SUB_COMPS_INTERFACE, and MTL_ITEM_REVISIONS_INTERFACE tables to import bill details. The Bill and Routing Interface program derives values for these tables the same way it does for the BOM_BILL_OF_MTLS_INTERFACE and BOM_INVENTORY_COMPS_INTERFACE tables.

You can assign standard components to any bill of material type. However, only standard components assigned to standard, model, and option class bills of material can have reference designators and substitute components. If you insert data in the BOM_REF_DESGS_INTERFACE or BOM_SUB_COMPS_INTERFACE tables for a planning bill, the Bill and Routing Interface program fails to import the record and sets the PROCESS_FLAG to **3** (Assign/Validation Failed).

As with the other tables, when you insert a record in the table, you must set the PROCESS_FLAG column to **1** (Pending).

Type of Bill Detail	Detail Interface Table	Parent Interface Table	Rows Per Parent Record
Reference Designators	BOM_REF_DESGS_INTERFACE	BOM_INVENTORY_COMPS_INTERFACE	1 or more
Substitute Components	BOM_SUB_COMPS_INTERFACE	BOM_INVENTORY_COMPS_INTERFACE	1 or more
Item Revisions	MTL_ITEM_REVISIONS_INTERFACE	BOM_BILL_OF_MTLS_INTERFACE	1 or more

Table 2 – 8 Bill of Material Details (Page 1 of 1)

Required Data

Creating Reference Designators

You can only import data into the BOM_REF_DESGS_INTERFACE table for standard components assigned to standard, model, and option class bills of material. You must assign values to the following columns when creating standard components:

- COMPONENT_REFERENCE_DESIGNATOR
- COMPONENT_SEQUENCE_ID
- PROCESS_FLAG
- TRANSACTION_TYPE

Updating Reference Designators

The following columns are required when updating data in the BOM_REF_DESGS_INTERFACE table:

- COMPONENT_SEQUENCE_ID

- COMPONENT_REFERENCE_DESIGNATOR

Deleting Reference Designators

The following columns are required when deleting data in the BOM_REF_DESGS_INTERFACE table:

- COMPONENT_SEQUENCE_ID
- COMPONENT_REFERENCE_DESIGNATOR

The specified record will be deleted upon import.

Attention: You cannot delete an unimplemented record.

Creating Substitute Components

You can only import data into the BOM_SUB_COMPS_INTERFACE table for standard components assign to standard, model, and option class bills of material. You must assign values to the following columns:

- SUBSTITUTE_COMPONENT_ID
- SUBSTITUTE_ITEM_QUANTITY
- COMPONENT_SEQUENCE_ID
- PROCESS_FLAG
- TRANSACTION_TYPE

Updating Substitute Components

The following columns are required when updating data in the BOM_SUB_COMPS_INTERFACE table:

- COMPONENT_SEQUENCE_ID
- SUBSTITUTE_COMP_NUMBER or SUBSTITUTE_COMP_NUMBER

Deleting Substitute Components

The following columns are required when deleting data in the BOM_SUB_COMPS_INTERFACE table:

- COMPONENT_SEQUENCE_ID
- SUBSTITUTE_COMP_NUMBER or SUBSTITUTE_COMP_NUMBER

The specified record will be deleted upon import.

Attention: You cannot delete an unimplemented record.

Creating Item Revisions

You can import data into the MTL_ITEM_REVISIONS_INTERFACE table to assign one or more item revision to your bill of material. You must assign values to the following columns:

- INVENTORY_ITEM_ID
- ORGANIZATION_ID
- REVISION
- PROCESS_FLAG
- TRANSACTION_TYPE

Updating Item Revisions

The following columns are required when updating data in the MTL_ITEM_REVISIONS_INTERFACE table:

- INVENTORY_ITEM_ID or ITEM_NUMBER
- ORGANIZATION_ID or ORGANIZATION_CODE
- REVISION

Attention: The Bill and Routing Interface program does not allow the deleting of item revisions.

Derived Data

After inserting data into the BOM_REF_DESGS_INTERFACE or BOM_SUB_COMPS_INTERFACE table, the Bill and Routing Interface program derives the value for the BILL_SEQUENCE_ID column if you assign values to the following columns:

- ASSEMBLY_ITEM_ID
- ORGANIZATION_ID
- ALTERNATE_BOM_DESIGNATOR

The program also assigns a value for the COMPONENT_SEQUENCE_ID column if you enter values into the following columns:

- BILL_SEQUENCE_ID

- COMPONENT_ITEM_ID
- OPERATION_SEQ_NUM
- EFFECTIVITY_DATE

Optional Data

Updating Reference Designators

The following columns in the BOM_REF_DESGS_INTERFACE table are updatable:

- NEW_DESIGNATOR
- REF_DESIGNATOR_COMMENT

Updating Substitute Components

The following columns in the BOM_SUB_COMPS_INTERFACE table are updatable:

- NEW_SUB_COMP_ID or NEW_SUB_COMP_NUMBER
- SUBSTITUTE_ITEM_QUANTITY

Updating Item Revisions

The following columns in the MTL_ITEM_REVISIONS_INTERFACE table are updatable:

- DESCRIPTION
- EFFECTIVITY_DATE

Attention: You cannot update an unimplemented record.

Validating Interface Table Rows

After you load the bill of material, product family, and component data, the Bill and Routing Interface program validates the required data for the five interface tables. Bill of material validation insures that each row has an included or defaulted value for all the required columns, and verifies any interdependent relationships. The program validates the data the same way as the Bills of Material form and the Engineering Bills of Material form validate manually entered bills. For example,

you cannot import a standard bill and assign model, option class, or planning items as components.

When you import bills, the Bill and Routing Interface program validates all rows in the BOM_BILL_OF_MTLS_INTERFACE, BOM_INVENTORY_COMPS_INTERFACE, BOM_REF_DESGS_INTERFACE, BOM_SUB_COMPS_INTERFACE, and MTL_ITEM_REVISIONS_INTERFACE tables that have a PROCESS_FLAG set to **1** (Pending).

If the Bill and Routing Interface program cannot assign a value to a row or validate that row, the program sets the PROCESS_FLAG for the row to **3** (Assign/Validation Failed) and inserts a row in the MTL_INTERFACE_ERRORS table. To identify the error message for a failed row, the program automatically populates the UNIQUE_ID column in the error interface table with the same value as the TRANSACTION_ID value. Each error also has a value for the MESSAGE_NAME and REQUEST_ID columns in the error interface table. The MESSAGE_NAME column corresponds to messages stored in Oracle Application Message Dictionary and the REQUEST_ID column stores the concurrent request id. If the program detects any internal database errors, the program stores the internal error in the MESSAGE_NAME column and stores the specific database error message in the ERROR_MESSAGE column.

When the Bill and Routing Interface successfully assigns values and validates rows, the program inserts the rows into the BOM_BILL_OF_MATERIALS, BOM_INVENTORY_COMPONENTS, and if appropriate, the BOM_REFERENCE_DESIGNATORS, BOM_SUBSTITUTE_COMPONENTS, and MTL_ITEM_REVISIONS tables for each validated row. If the program cannot insert the row into one of these production tables, the program sets the PROCESS_FLAG for that row to **4** (Import Failed).

After the Bill and Routing Interface program successfully creates, updates or deletes a bill of material and components in the production tables, it sets the PROCESS_FLAG column to **7** (Import Succeeded) for the successfully processed rows. You can optionally delete all the successfully processed rows from the interface tables, after the records have been imported.

If you import a bill of material with multiple components and one of the components fails validation, the Bill and Routing Interface program imports the bill of material and the other validated components. The program only fails the record where the error occurred. If however, the row in the BOM_BILL_OF_MTLS_INTERFACE table fails, the bill of material and product family and all of its details are not imported. If

you insert reference designators and substitute component information directly into the BOM_REF_DESGS_INTERFACE and BOM_SUB_COMPS_INTERFACE tables, all records for that table must succeed validation for the data to be imported. For example, if you are importing multiple reference designators into the BOM_REF_DESGS_INTERFACE table and one fails validation, none of the other records are imported.

See Also

Bills of Material, *Oracle Bills of Material User's Guide*

Engineering Prototype Environment, *Oracle Engineering User's Guide*

Define Messages, *Oracle Application Object Library Reference Manual*

Resolving Failed Interface Table Rows

Reviewing Failed Rows

You can review and report rows in the interface tables using SQL*Plus or any custom reports you develop. Since all rows in the interface table have a value for PROCESS_FLAG, you can easily identify records that are successfully imported into Oracle Bills of Material and Oracle Engineering, or records that failed validation or import. You can also identify individual records by the unique value for the TRANSACTION_ID column.

Correcting Failed Rows

You can update any row from the interface tables using SQL*Plus. If you update a row to resolve invalid data, you must set the PROCESS_FLAG to **1** (Pending) for that row. If you delete a failed row and insert a replacement row, you should set the PROCESS_FLAG to **1** (Pending) for the new row. When you resubmit the Bill and Routing Interface program all rows pending validation are processed.

Open Routing Interface

You can automatically import routings into Oracle Bills of Material or Oracle Engineering from any source using the Bill and Routing Interface. With this interface, you can easily convert manufacturing routings from legacy manufacturing systems or import new engineering routings from Product Data Management (PDM) systems. This interface also allows you to update or delete existing routings. Oracle Bills of Material and Oracle Engineering validate your data, ensuring that your imported routings contain the same routing details as those you enter manually in the Define Routing or the Define Engineering Routing form.

The purpose of this essay is to explain how to use the Bill and Routing Interface so you can seamlessly integrate other applications with Oracle Bills of Material and Oracle Engineering.

See Also

Routings, *Oracle Bills of Material User's Guide*

Engineering Prototype Environment, *Oracle Engineering User's Guide*

Functional Overview

Once you install Oracle Bills of Material and Oracle Engineering, you can use the Bill and Routing Interface program to update, delete, or create new manufacturing and engineering routings. The Bill and Routing Interface validates your data the same way Oracle Bills of Material and Oracle Engineering verify routings entered manually.

Before you use the Bill and Routing Interface, you must write and run a custom program that extracts routing, operation, and resource details from your source system. This program must insert rows in the following tables for each extracted routing:

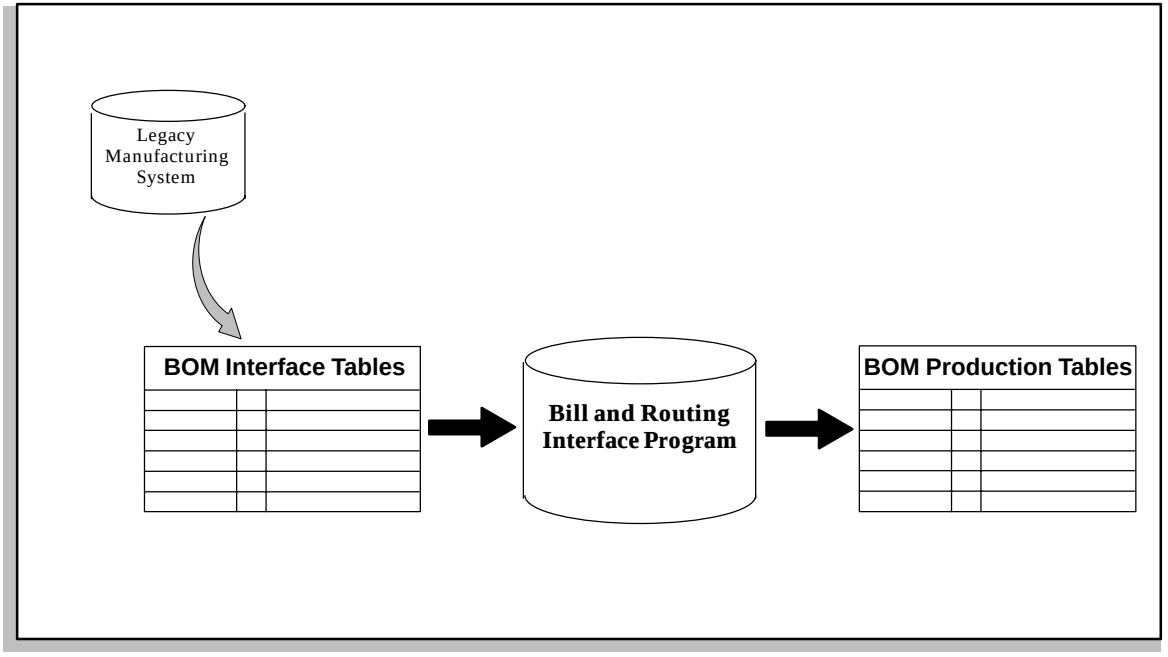
- BOM_OP_ROUTINGS_INTERFACE
- BOM_OP_SEQUENCES_INTERFACE
- BOM_OP_RESOURCES_INTERFACE
- MTL_RTG_ITEM_REVS_INTERFACE

After you load the data in the interface tables, you can launch the Bill and Routing Interface program from the Import Bills and Routings

form in Oracle Bills of Material or Oracle Engineering. The program assigns values, validates the data you include, and then imports the new routings or updates.

The following diagram displays the import process, from inserting data into the interface tables, through importing data into the Oracle Bills of Material and Oracle Engineering production tables.

Figure 2 – 3



See Also

Routings, *Oracle Bills of Material User's Guide,*

Define Routing, *Oracle Bills of Material User's Guide*

Engineering Prototype Environment, *Oracle Engineering User's Guide*

Creating a Routing

To create a routing you must enter "Insert" in the TRANSACTION_TYPE column. To import a routing with operations and resources, you must populate the BOM_OP_ROUTINGS_INTERFACE, BOM_OP_SEQUENCES_INTERFACE, and

BOM_OP_RESOURCES_INTERFACE tables. Using these three tables, you can create routing header information and assign operation and resource details.

You can optionally create a routing revision when you import a routing, by inserting a revision at the same time you insert your routing data. If you enter a value in the PROCESS_REVISION column of the BOM_OP_ROUTINGS_INTERFACE table, the Bill and Routing Interface program inserts a row into the MTL_RTG_ITEM_REVS_INTERFACE table. To insert multiple routing revisions, you can insert data directly into the MTL_RTG_ITEM_REVS_INTERFACE table. If the validation is successful, the program then imports the data into the MTL_RTG_ITEM_REVISIONS table.

Updating a Routing

To update a routing you must enter ”Update” in the TRANSACTION_TYPE column. To identify the routing that you would like to update you must enter the required values in the BOM_OP_ROUTINGS_INTERFACE, BOM_OP_SEQUENCES_INTERFACE, and BOM_OP_RESOURCES_INTERFACE tables.

If a column is left blank, the program interprets the blank value as do not update. To update a column to null you must enter the following values:

Data Type	Value
Null Character	”empty” (case sensitive)
Null Date	01–JAN–1900 at midnight
Null Number	–999999

If you are using SQL Plus, you can use the following packaged constants:

- BOM_RoutingInterface_PUB.G_NullChar
- BOM_RoutingInterface_PUB.G_NullNum
- BOM_RoutingInterface_PUB.G_NullDate

Deleting a Routing

To delete a routing you must enter ”Delete” in the TRANSACTION_TYPE column. To delete operations or routings, you must enter unique delete group names for routings or operations.

Validation

After populating the interface tables, you can run the Bill and Routing Interface program. The column, PROCESS_FLAG, indicates the current state of processing for a row in the interface table. Possible values for the column include:

1 – Pending

3 – Import Failed

7 – Import Succeeded

When you insert, update, or delete records, you must set the PROCESS_FLAG to **1** (Pending). The program assigns and validates all rows with a status of **1** (Pending) and then imports them into the production tables. If the assign, validate, or transact procedure fails for a row, the program sets the PROCESS_FLAG to **3** (Import Failed) for that row. Successfully imported rows have a value of **7** (Import Succeeded) for the PROCESS_FLAG.

When you submit the Bill and Routing Interface program, Oracle Bills of Material or Oracle Engineering automatically updates the TRANSACTION_ID columns in each of the interface tables. The column TRANSACTION_ID stores a unique id for each row in the interface table.

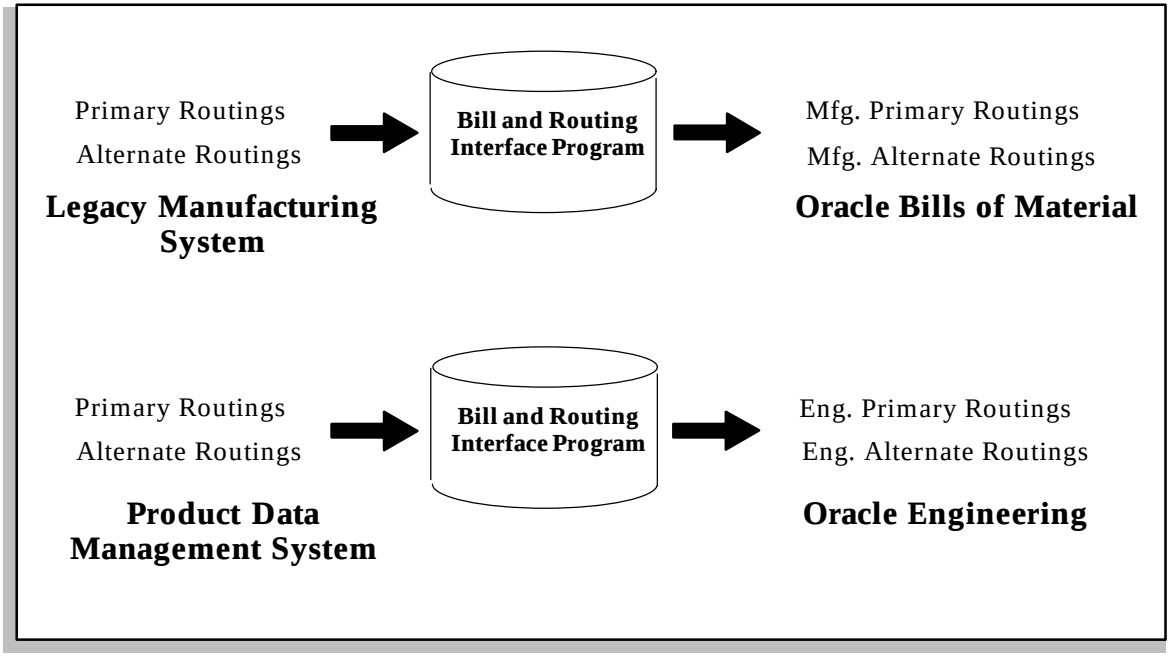
Although you can import bills and routings simultaneously, all routing operations must exist before you can assign a component to an operation. If a routing does not exist, you cannot assign an operation sequence to a component on a bill of material.

You can simultaneously create primary and alternate routings. Since the Bill and Routing Interface program validates data the same way the Define Routing or Define Engineering Routing form verifies data, you cannot define an alternate routing if the primary routing does not exist.

Specify a value in the ROUTING_TYPE column to determine the type of routing you want to create. If you create a manufacturing routing, the routing is visible to both Oracle Bills of Material and Oracle Engineering. However, if you create an engineering routing, it is only accessible through the Oracle Engineering product. The following diagram illustrates the data the Bill and Routing interface program

processes and imports into Oracle Bills of Material and Oracle Engineering.

Figure 2 – 4



See Also

Bills of Material, *Oracle Bills of Material User's Guide*

Routings, *Oracle Bills of Material User's Guide*

Define Routing, *Oracle Bills of Material User's Guide*

Oracle Bills of Material Technical Reference Manual

Setting Up for Routing Import

When you create, update, or delete routings, there are no additional setup steps in Oracle Bills of Material or Oracle Engineering beyond those already required to manually define manufacturing and engineering routings. Standard operations must be defined in Oracle Bills of Material before they can be assigned to a manufacturing or engineering routing. Since you launch and manage the Bill and

Routing Interface program through the concurrent manager, you must insure that the concurrent manager is running before you can create, update, or delete any routings.

For each ”Production” table there exists a corresponding”Interface” table:

Production Table	Interface Table
BOM_OPERATIONAL_ROUTINGS	BOM_OP_ROUTINGS_INTERFACE
MTL_RTG_ITEM_REVISIONS	MTL_RTG_ITEM_REVS_INTERFACE
BOM_OPERATION_SEQUENCES	BOM_OP_SEQUENCES_INTERFACE
BOM_OPERATION_RESOURCES	BOM_OP_RESOURCES_INTERFACE

Table 2 – 9 Routing Production and Interface tables (Page 1 of 1)

The ”Interface” tables contain all the columns that their ”Production” tables have, in addition to TRANSACTION_ID, PROCESS_FLAG, TRANSACTION_TYPE, and user–friendly columns. The TRANSACTION_ID column uniquely identifies a record. The PROCESS_FLAG column keeps track of where the record is in the Open Interface process. The TRANSACTION_TYPE column contains the values: ”Insert,” ”Update,” or ”Delete.”

See Also

Routings, *Oracle Bills of Material User’s Guide*

Setting Up Oracle Bills of Material, *Oracle Bills of Material User’s Guide*

Setting Up Oracle Engineering, *Oracle Bills of Material User’s Guide*

Bill and Routing Interface Runtime Options

When you run the Bill and Routing Interface program, you must specify a number of runtime options. These options include:

All Organizations

Yes Run the interface for all organization codes in the bill of material and routing interface tables.

No Run the interface program only for the organization you are currently in. The interface program only processes bill of material and routing interface records in your current organization.

Attention: When you specify **No** for this option, the program ignores rows in the interface tables that do not have an organization code or organization id assigned.

Import Routings

Yes Import records from the routing interface tables for your current organization or all organizations.

No Do not import records from the routing interface tables.

Import Bills of Material

Yes Import records from the bills of material interface tables for your current organization or all organizations.

No Do not import records from the bills of material interface tables.

Delete Processed Rows

Yes Delete successfully processed rows from the bill of material and routing interface tables.

No Leave all records in the bill of material and routing interface tables for successfully processed rows.

Inserting into BOM_OP_ROUTINGS_INTERFACE Table

You must load the routing information you want to import from your source system into the BOM_OP_ROUTINGS_INTERFACE, BOM_OP_SEQUENCES_INTERFACE, and BOM_OP_RESOURCES_INTERFACE tables. Oracle Bills of Material defaults any additional data, validates your data, and then creates, updates, or deletes manufacturing or engineering routings in Oracle Bills of Material or Oracle Engineering.

BOM_OP_ROUTINGS_INTERFACE Table Description

The following graphic describes the columns in the BOM_OP_ROUTINGS_INTERFACE table that you use to insert data.

BOM_OP_ROUTINGS_ INTERFACE Column Name	Type	Derived or Defaulted Value in Production Tables
ROUTING_SEQUENCE_ID	Number	Sequence BOM_OPERATIONAL_ ROUTINGS_S
ASSEMBLY_ITEM_ID	Number	From ASSEMBLY_ITEM_NUMBER
ORGANIZATION_ID	Number	From ORGANIZATION_CODE
ALTERNATE_ROUTING_DESIGNATOR	Varchar (10)	
LAST_UPDATE_DATE	Date	System Date
LAST_UPDATED_BY	Number	Userid
CREATION_DATE	Date	System Date
CREATED_BY	Number	Userid
ROUTING_TYPE	Number	1
COMMON_ASSEMBLY_ITEM_ID	Number	From COMMON_ITEM_NUMBER
COMMON_ROUTING_SEQUENCE_ID	Number	Sequence
TRANSACTION_TYPE	Varchar2(4)	
ROUTING_COMMENT	Varchar (240)	
COMPLETION_SUBINVENTORY	Varchar (10)	
COMPLETION_LOCATOR_ID	Number	
LINE_ID	Number	
LINE_CODE	Number	
CFM_ROUTING_FLAG	Number	
MIXED_MODEL_MAP_FLAG	Number	
CTP_FLAG	Number	
TOTAL_PRODUCT_CYCLE_TIME	Number	
PRIORITY	Number	

Table 2 – 10 BOM Routing Interface (Page 1 of 2)

BOM_OP_ROUTINGS_ INTERFACE Column Name	Type	Derived or Defaulted Value in Production Tables
ATTRIBUTE_CATEGORY	Varchar (30)	
ATTRIBUTE1	Varchar (150)	
ATTRIBUTE2	Varchar (150)	
ATTRIBUTE3	Varchar (150)	
ATTRIBUTE4	Varchar (150)	
ATTRIBUTE5	Varchar (150)	
ATTRIBUTE6	Varchar (150)	
ATTRIBUTE7	Varchar (150)	
ATTRIBUTE8	Varchar (150)	
ATTRIBUTE9	Varchar (150)	
ATTRIBUTE10	Varchar (150)	
ATTRIBUTE11	Varchar (150)	
ATTRIBUTE12	Varchar (150)	
ATTRIBUTE13	Varchar (150)	
ATTRIBUTE14	Varchar (150)	
ATTRIBUTE15	Varchar (150)	
PROCESS_REVISION	Varchar (3)	
ORGANIZATION_CODE	Varchar (3)	
ASSEMBLY_ITEM_NUMBER	Varchar (81)	
COMMON_ITEM_NUMBER	Varchar (81)	
LOCATION_NAME	Varchar (81)	
PROCESS_FLAG	Number	
REQUEST_ID	Number	From FND_CONCURRENT_REQUESTS

Table 2 – 10 BOM Routing Interface (Page 2 of 2)

Attention: It is only necessary to insert a value for the ALTERNATE_ROUTING_DESIGNATOR column when you

import an **alternate** routing. And, it is only necessary to insert a value for the COMMON_ASSEMBLY_ITEM_ID or COMMON_ROUTING_SEQUENCE_ID columns when the routing you import references a **common** routing.

Attention: You cannot dynamically create locations for your subinventories using the interface tables. If you specify a value for the COMPLETION_LOCATOR_ID column, the location must already exist in Oracle Inventory.

Required Data

Creating a Routing

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

- PROCESS_FLAG (1)
- ASSEMBLY_ITEM_ID
- ORGANIZATION_ID
- ROUTING_TYPE
- TRANSACTION_TYPE (Insert)

If you are creating an alternate routing you must also enter a value in the ALTERNATE_ROUTING_DESIGNATOR column.

If the routing you import references a common routing, you must enter a value in the COMMON_ASSEMBLY_ITEM_ID or the COMMON_ROUTING_SEQUENCE_ID column. Routings can only reference common routings that belong to the same organization. If the routing does not reference a common routing, the Bill and Routing Interface program defaults the value of the ROUTING_SEQUENCE_ID for the COMMON_ROUTING_SEQUENCE_ID.

You can specify in the ROUTING_TYPE column whether the routing is a manufacturing routing or an engineering routing. If you do not include a value for this column, Oracle Bills of Material defaults a value of **1** (manufacturing), and creates a manufacturing routing. To create an engineering routing, you must enter a value of **2** (engineering) for the ROUTING_TYPE column.

When you insert new rows into the BOM_OP_ROUTINGS_INTERFACE table, you must set the PROCESS_FLAG to **1** (Pending).

Updating a Routing

To identify the routing record you are trying to update, you must provide one of the following:

- ROUTING_SEQUENCE_ID or
- ASSEMBLY_ITEM_ID, ORGANIZATION_ID, and ALTERNATE_ROUTING_DESIGNATOR

You must always enter values for the following required columns when you update rows into the BOM_OP_ROUTINGS_INTERFACE table.

- TRANSACTION_TYPE (Update)
- PROCESS_FLAG (1)

Deleting a Routing

To identify the routing record you are trying to delete, you must provide one of the following:

- ROUTING_SEQUENCE_ID or
- ASSEMBLY_ITEM_ID, ORGANIZATION_ID, AND ALTERNATE_ROUTING_DESIGNATOR

You must always enter values for the following required columns when you update rows into the BOM_OP_ROUTINGS_INTERFACE table.

- TRANSACTION_TYPE (Delete)
- PROCESS_FLAG (1)

Derived Data

Creating a Routing

The Bill and Routing Interface program derives or defaults most of the data required to create a manufacturing or an engineering routing. The Bill and Routing Interface program derives or defaults the columns using the same logic as the Define Routing form or the Define Engineering Routing form. When you populate a column in the interface table, the program imports the row with the data you include and does not default a value. However, if you do not enter data in a derived or defaulted column, the program automatically imports the row with the derived or defaulted value. For example, if you do not specify a value for the ROUTING_TYPE column, the Bill and Routing Interface program automatically defaults to 1, a manufacturing routing.

The BOM_OP_ROUTINGS_INTERFACE table contains user–friendly columns that let you easily enter data in the interface table. The Bill and Routing Interface program uses the values you enter for these columns to derive unique identifiers. For example, when you import a routing you can include a value for ASSEMBLY_ITEM_NUMBER or ASSEMBLY_ITEM_ID. If you enter a value for ASSEMBLY_ITEM_NUMBER, the Bill and Routing Interface program derives the value for ASSEMBLY_ITEM_ID. The following table lists these columns and the corresponding column that stores the derived value:

BOM_OP_ROUTINGS_INTERFACE User–Friendly Column Name	Derived Column Name
ASSEMBLY_ITEM_NUMBER	ASSEMBLY_ITEM_ID
ORGANIZATION_CODE	ORGANIZATION_ID
COMMON_ITEM_NUMBER	COMMON_ASSEMBLY_ITEM_ID
LOCATION_NAME	COMPLETION_LOCATOR_ID
LINE_CODE	LINE_ID

Table 2 – 11 BOM Routing Interface, Derived Columns (Page 1 of 1)

Attention: If you enter a value for the ASSEMBLY_ITEM_NUMBER or COMMON_ITEM_NUMBER column, you must insert the system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for an item, it searches for this separator to indicate the end of one segment value and the start of the next segment value. See: Defining Key Flexfield Segments, *Oracle Applications Flexfields Guide*

The Bill and Routing Interface program derives values for some columns in BOM_OP_ROUTINGS_INTERFACE based on values entered for other columns. For example, if you insert a value in the COMMON_ASSEMBLY_ITEM_ID column, the program derives the value for the COMMON_ROUTING_SEQUENCE_ID column.

Updating a Routing

The Bill and Routing Interface program does not derive or default data when updating a routing.

Deleting a Routing

The Bill and Routing Interface program does not derive or default data when deleting a routing.

See Also

Oracle Bills of Material Technical Reference Manual

Oracle Engineering Technical Reference Manual

Optional Data

Creating a Routing

You can optionally assign a value to the other columns in the interface table or let the Bill and Routing Interface program default a value.

You can insert a value in the PROCESS_REVISION column to import one routing revision. The Bill and Routing Interface program inserts a row into the MTL_RTG_ITEM_REVS_INTERFACE table, validates the data and then imports the record into the MTL_RTG_ITEM_REVISIONS table. To insert multiple revisions for a routing, you must insert the data directly into the MTL_RTG_ITEM_REVS_INTERFACE table. See: Importing Additional Routing Information: page 2 – 66

You can also populate the descriptive flexfield segment columns in the interface table to import values for the routing descriptive flexfield. You define the structure for the descriptive flexfield in the column ATTRIBUTE_CATEGORY and you can populate each segment value in the columns ATTRIBUTE1 through ATTRIBUTE15. After importing a new routing, you can view your descriptive flexfield values in the Define Routing form.

Updating a Routing

The optional data for updating a routing is the same as creating a routing in the BOM_OP_ROUTINGS_INTERFACE table.

Deleting a Routing

Once the routing is identified the entire record will be placed in a delete group. There is no optional data.

Inserting into BOM_OP_SEQUENCES_INTERFACE Table

You must load the operation details for each routing in the BOM_OP_SEQUENCES_INTERFACE table. Oracle Bills of Material and Oracle Engineering use this information, along with the data in the BOM_OP_ROUTINGS_INTERFACE and BOM_OP_RESOURCES_INTERFACE tables, to create new routings.

BOM_OP_SEQUENCES_INTERFACE Table Description

The following graphic describes the columns in the BOM_OP_SEQUENCES_INTERFACE table that you use to insert data.

BOM_OP_SEQUENCES_ INTERFACE Column Name	Type	Derived or Defaulted Value in Production Tables
OPERATION_SEQUENCE_ID	Number	Sequence BOM_OPERATION_ SEQUENCES_S
ROUTING_SEQUENCE_ID	Number	From BOM_OPERATIONAL_ROUTINGS
OPERATION_SEQ_NUM	Number	
NEW_OPERATION_SEQ_NUM	Number	
LAST_UPDATE_DATE	Date	System Date
LAST_UPDATED_BY	Number	Userid
CREATION_DATE	Date	System Date
CREATED_BY	Number	Userid
STANDARD_OPERATION_ID	Number	From OPERATION_CODE
DEPARTMENT_ID	Number	From DEPARTMENT_CODE
TRANSACTION_TYPE	Varchar2 (10)	
OPERATION_LEAD_TIME_PERCENT	Number	
MINIMUM_TRANSFER_QUANTITY	Number	0
COUNT_POINT_TYPE	Number	1
OPERATION_DESCRIPTION	Varchar (240)	
EFFECTIVITY_DATE	Date	System Date

Table 2 – 12 BOM Operation Sequences Interface (Page 1 of 4)

BOM_OP_SEQUENCES_INTERFACE Column Name	Type	Derived or Defaulted Value in Production Tables
NEW_EFFECTIVITY_DATE	Date	From EFFECTIVITY_DATE
ASSEMBLY_TYPE	Number	
DISABLE_DATE	Date	
BACKFLUSH_FLAG	Number	1
OPERATION_TYPE	Number	
REFERENCE_FLAG	Number	
PROCESS_OP_SEQ_ID	Number	
LINE_OP_SEQ_ID	Number	
YIELD	Number	
CUMULATIVE_YIELD	Number	
REVERSE_CUMULATIVE_YIELD	Number	
LABOR_TIME_CALC	Number	
MACHINE_TIME_CALC	Number	
TOTAL_TIME_CALC	Number	
LABOR_TIME_USER	Number	
MACHINE_TIME_USER	Number	
TOTAL_TIME_USER	Number	
NET_PLANNING_PERCENT	Number	
ATTRIBUTE_CATEGORY	Varchar (30)	
ATTRIBUTE1	Varchar (150)	
ATTRIBUTE2	Varchar (150)	
ATTRIBUTE3	Varchar (150)	
ATTRIBUTE4	Varchar (150)	
ATTRIBUTE5	Varchar (150)	
ATTRIBUTE6	Varchar (150)	
ATTRIBUTE7	Varchar (150)	

Table 2 – 12 BOM Operation Sequences Interface (Page 2 of 4)

BOM_OP_SEQUENCES_INTERFACE Column Name	Type	Derived or Defaulted Value in Production Tables
ATTRIBUTE8	Varchar (150)	
ATTRIBUTE9	Varchar (150)	
ATTRIBUTE10	Varchar (150)	
ATTRIBUTE11	Varchar (150)	
ATTRIBUTE12	Varchar (150)	
ATTRIBUTE13	Varchar (150)	
ATTRIBUTE14	Varchar (150)	
ATTRIBUTE15	Varchar (150)	
REQUEST_ID	Number	From FND_CONCURRENT_REQUESTS
ASSEMBLY_ITEM_ID	Number	From ASSEMBLY_ITEM_NUMBER
OPTION_DEPENDENT_FLAG	Number	2
ORGANIZATION_ID	Number	From ORGANIZATION_CODE
ALTERNATE_ROUTING_DESIGNATOR	Varchar (10)	
ORGANIZATION_CODE	Varchar (3)	
ASSEMBLY_ITEM_NUMBER	Varchar (81)	
DEPARTMENT_CODE	Varchar (10)	
OPERATION_CODE	Varchar (4)	
RESOURCE_ID1	Number	From RESOURCE_CODE1
RESOURCE_ID2	Number	From RESOURCE_CODE2
RESOURCE_ID3	Number	From RESOURCE_CODE3
RESOURCE_CODE1	Varchar (10)	
RESOURCE_CODE2	Varchar (10)	
RESOURCE_CODE3	Varchar (10)	
INSTRUCTION_CODE1	Varchar (10)	
INSTRUCTION_CODE2	Varchar (10)	

Table 2 – 12 BOM Operation Sequences Interface (Page 3 of 4)

BOM_OP_SEQUENCES_INTERFACE Column Name	Type	Derived or Defaulted Value in Production Tables
INSTRUCTION_CODE3	Varchar (10)	
PROCESS_FLAG	Number	

Table 2 – 12 BOM Operation Sequences Interface (Page 4 of 4)

Attention: You must fill in ORGANIZATION_ID or ORG_CODE if you want to import records for a specific organization and not for all organizations.

Attention: It is only necessary to insert a value for the ALTERNATE_ROUTING_DESIGNATOR column when you import an **alternate** routing.

The following columns apply to the Oracle Flow Manufacturing product only. If you are not using Oracle Flow Manufacturing, you should not put values in these columns.

- OPERATION_TYPE
- PROCESS_OPSEQ_ID
- LINE_OP_SEQ_ID
- CUMULATIVE_YIELD
- REVERSE_CUMULATIVE_YIELD
- LABOR_TIME_CALC
- MACHINE_TIME_CALC
- TOTAL_TIME_CALC
- LABOR_TIME_USER
- MACHINE_TIME_USER
- TOTAL_TIME_USER
- NET_PLANNING_PERCENT

Required Data

Creating a Operation

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

Each imported record must have a value for the following columns.

- PROCESS_FLAG (1)
- ROUTING_SEQUENCE_ID
- OPERATION_SEQ_NUM
- DEPARTMENT_ID
- EFFECTIVITY_DATE
- TRANSACTION_TYPE (Insert)

You also must specify a value in the ALTERNATE_ROUTING_DESIGNATOR column if you assign operations to an alternate routing and have not entered a value for the ROUTING_SEQUENCE_ID column.

Updating a Operation

To identify the operation record you are trying to update, you must provide one of the following:

- OPERATION_SEQUENCE_ID or
- ROUTING_SEQUENCE_ID, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, OPERATION_TYPE or
- ASSEMBLY_ITEM_ID, ASSEMBLY_ITEM_NUMBER, ALTERNATE_ROUTING_DESIGNATOR, ORGANIZATION_ID, ORGANIZATION_CODE, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, and OPERATION_TYPE

Each imported record must have a value for the following columns.

- TRANSACTION_TYPE (Update)
- PROCESS_FLAG (1)

To update the OPERATION_SEQ_NUM or EFFECTIVITY_DATE, you must enter a value in the NEW_OP_SEQ_NUM and NEW_EFFECTIVITY_DATE columns.

Deleting a Operation

To identify the operation record you are trying to delete, you must provide one of the following:

- OPERATION_SEQUENCE_ID or
- ROUTING_SEQUENCE_ID, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, OPERATION_TYPE or
- ASSEMBLY_ITEM_ID, ASSEMBLY_ITEM_NUMBER, ALTERNATE_ROUTING_DESIGNATOR, ORGANIZATION_ID, ORGANIZATION_CODE, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, OPERATION_TYPE

Each imported record must have a value for the following columns.

- TRANSACTION_TYPE (Delete)

Derived Data

Creating a Operation

The Bill and Routing Interface program derives or defaults most of the data required to assign operations to a routing. You can optionally include a value for any derived or defaulted column, as well as data for any of the other columns in the interface table.

The Interface program uses the same logic to derive or default column values in the BOM_OP_SEQUENCES_INTERFACE table as in the BOM_OP_ROUTINGS_INTERFACE table; when you populate a column in the interface table, the Bill and Routing Interface program imports the row with the data you include. However, if you do not enter data in a derived or defaulted column, the program automatically imports the row with the derived or defaulted value.

The BOM_OP_SEQUENCES_INTERFACE table contains user–friendly columns that let you easily enter data in the interface table. The Bill and Routing Interface program uses the values you enter for these columns to derive unique identifiers. The following table lists these columns and the corresponding column that stores the derived value.

BOM_OP_SEQUENCES_INTERFACE User–Friendly Column Name	Derived Column Name
OPERATION_CODE	STANDARD_OPERATION_ID

BOM_OP_SEQUENCES_INTERFACE User–Friendly Column Name	Derived Column Name
DEPARTMENT_CODE	DEPARTMENT_ID
ORGANIZATION_CODE	ORGANIZATION_ID
ASSEMBLY_ITEM_NUMBER	ASSEMBLY_ITEM_ID
RESOURCE_CODE1	RESOURCE_ID1
RESOURCE_CODE2	RESOURCE_ID2
RESOURCE_CODE3	RESOURCE_ID3

Table 2 – 13 BOM Operation Sequences Interface, Derived Columns (Page 2 of 2)

Attention: If you enter a value for the ASSEMBLY_ITEM_NUMBER column, you must insert the system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for the item, it searches for this separator to indicate the end of one segment value and the start of the next segment value.

Although the ROUTING_SEQUENCE_ID and OPERATION_SEQUENCE_ID are required columns, the Bill and Routing Interface program derives values for the columns when you do not specify a value. The program derives a value for the ROUTING_SEQUENCE_ID if you insert data into the following columns:

- ASSEMBLY_ITEM_ID
- ORGANIZATION_ID
- ALTERNATE_ROUTING_DESIGNATOR

The OPERATION_SEQUENCE_ID is derived from the sequence, BOM_OPERATION_SEQUENCES_S.

If you enter a value for the STANDARD_OPERATION_ID column, the Bill and Routing Interface program defaults the values for the other columns based on the information defined in the standard operation. The program only defaults values for the columns that you have not already specified a value. To assign standard operations, you must have previously defined them in the Define Standard Operations form in Oracle Bills of Material. Resources and attachments are also

defaulted if you insert a value in the STANDARD_OPERATION_ID column. Resources must also be predefined in Oracle Bills of Material.

For a detailed description of the derived or defaulted columns, please refer to the Oracle Bills of Material Technical Reference Manual.

Updating a Operation

The Bill and Routing Interface program does not derive or default data when updating a routing.

Deleting a Operation

The Bill and Routing Interface program does not derive or default data when deleting a routing.

See Also

Defining Key Flexfield Segments, *Oracle Applications Flexfields Guide*

Oracle Bills of Material Technical Reference Manual

Oracle Engineering Technical Reference Manual

Optional Data

Creating a Operation

You can import three resources for each operation using the BOM_OP_SEQUENCES_INTERFACE table. Resources must be defined in Oracle Bills of Material before they can be assigned to manufacturing or engineering routings. To import more than three resources, you must insert rows directly in the BOM_OP_RESOURCES_INTERFACE table. And, if you insert resources using the BOM_OP_SEQUENCES_INTERFACE table, you must accept the default values that the Bill and Routing Interface program derives for the resource columns.

You can also populate the descriptive flexfield segment columns in the interface table to import values for the operation descriptive flexfield. You define the structure for the descriptive flexfield in the column ATTRIBUTE_CATEGORY and you can populate each segment value in the columns ATTRIBUTE1 through ATTRIBUTE15. After importing a routing, you can view your descriptive flexfield values in the Operations block of the Define Routing form.

Updating a Operation

The optional data for updating a routing is the same as creating a routing in the BOM_OP_SEQUENCES_INTERFACE table.

Deleting a Operation

Once the operation is identified the entire record will be placed in a delete group. There is no optional data.

Inserting into BOM_OP_RESOURCES_INTERFACE Table

You can load the resource details for each operation in the BOM_OP_RESOURCES_INTERFACE table. Oracle Bills of Material and Oracle Engineering use this resource information, along with the data in the BOM_OP_ROUTINGS_INTERFACE and BOM_OP_SEQUENCES_INTERFACE tables, to create new routings. Before you can assign resources to an operation, you must define the resources in the Define Resource form in Oracle Bills of Material.

See Also

Define Routing, *Oracle Bills of Material User's Guide*

BOM_OP_RESOURCES_INTERFACE Table Description

The following graphic describes the columns in the BOM_OP_RESOURCES_INTERFACE table that you use to insert data.

BOM_OP_RESOURCES_ INTERFACE Column Name	Type	Derived or Defaulted in Production Tables
OPERATION_SEQUENCE_ID	Number	From BOM_OPERATION_ SEQUENCES
RESOURCE_SEQ_NUM	Number	
NEW_RESOURCE_SEQ_NUM	Number	
RESOURCE_ID	Number	From RESOURCE_CODE

Table 2 – 14 BOM Resources Interface (Page 1 of 3)

BOM_OP_RESOURCES_ INTERFACE Column Name	Type	Derived or Defaulted in Production Tables
ACTIVITY_ID	Number	From ACTIVITY
STANDARD_RATE_FLAG	Number	From BOM_RESOURCES. STANDARD_RATE_FLAG
ASSIGNED_UNITS	Number	1
USAGE_RATE_OR_AMOUNT	Number	1
USAGE_RATE_OR_AMOUNT_INVERSE	Number	1
BASIS_TYPE	Number	From BOM_RESOURCES. DEFAULT_BASIS
SCHEDULE_FLAG	Number	2
LAST_UPDATE_DATE	Date	System Date
LAST_UPDATED_BY	Number	Userid
CREATION_DATE	Date	System Date
CREATED_BY	Number	Userid
TRANSACTION_TYPE	Varchar2 (10)	
RESOURCE_OFFSET_PERCENT	Number	
AUTOCHARGE_TYPE	Number	From BOM_RESOURCES. AUTOCHARGE_TYPE
ATTRIBUTE_CATEGORY	Varchar (30)	
ATTRIBUTE1	Varchar (150)	
ATTRIBUTE2	Varchar (150)	
ATTRIBUTE3	Varchar (150)	
ATTRIBUTE4	Varchar (150)	
ATTRIBUTE5	Varchar (150)	
ATTRIBUTE6	Varchar (150)	
ATTRIBUTE7	Varchar (150)	
ATTRIBUTE8	Varchar (150)	

Table 2 – 14 BOM Resources Interface (Page 2 of 3)

BOM_OP_RESOURCES_ INTERFACE Column Name	Type	Derived or Defaulted in Production Tables
ATTRIBUTE9	Varchar (150)	
ATTRIBUTE10	Varchar (150)	
ATTRIBUTE11	Varchar (150)	
ATTRIBUTE12	Varchar (150)	
ATTRIBUTE13	Varchar (150)	
ATTRIBUTE14	Varchar (150)	
ATTRIBUTE15	Varchar (150)	
REQUEST_ID	Number	From FND_CONCURRENT_ REQUESTS
ASSEMBLY_ITEM_ID	Number	From ASSEMBLY_ITEM_NUMBER
ALTERNATE_ROUTING_DESIGNATOR	Varchar (10)	
ORGANIZATION_ID	Number	From ORGANIZATION_CODE
OPERATION_SEQ_NUM	Number	
EFFECTIVITY_DATE	Date	
ROUTING_SEQUENCE_ID	Number	
ORGANIZATION_CODE	Varchar (3)	
ASSEMBLY_ITEM_NUMBER	Varchar (81)	
RESOURCE_CODE	Varchar (10)	
ACTIVITY	Varchar (10)	
PROCESS_FLAG	Number	

Table 2 – 14 BOM Resources Interface (Page 3 of 3)

Attention: It is only necessary to insert a value for the ALTERNATE_ROUTING_DESIGNATOR column when you import an **alternate** routing, and you do not enter a value for the ROUTING_SEQUENCE_ID and OPERATION_SEQUENCE_ID columns.

Required Data

Creating a Resource

You must include data for the following columns in the BOM_OP_RESOURCES_INTERFACE table:

- PROCESS_FLAG (1)
- RESOURCE_SEQ_NUM
- RESOURCE_ID
- OPERATION_SEQUENCE_ID
- TRANSACTION_TYPE (Insert)

You also must specify a value in the ALTERNATE_ROUTING_DESIGNATOR column if you assign resources to an alternate routing and have not entered a value for the ROUTING_SEQUENCE_ID or the OPERATION_SEQUENCE_ID column.

Updating a Resource

To identify the resource record you are trying to update, you must provide one of the following:

- OPERATION_SEQUENCE_ID, RESOURCE_SEQ_NUM or
- ROUTING_SEQUENCE_ID, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, RESOURCE_SEQ_NUM or
- ASSEMBLY_ITEM_ID, ASSEMBLY_ITEM_NUMBER, ALTERNATE_ROUTING_DESIGNATOR, ORGANIZATION_ID, ORGANIZATION_CODE, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, and RESOURCE_SEQ_NUM

Each imported record must have a value for the following columns.

- TRANSACTION_TYPE (Update)
- PROCESS_FLAG (1)

To update the RESOURCE_SEQ_NUM, you must enter a value in the NEW_RESOURCE_SEQ_NUM column. The RESOURCE_SEQ_NUM column is used to determine which record will be updated.

Deleting a Resource

To identify the resource record you are trying to delete, you must provide one of the following:

- OPERATION_SEQUENCE_ID, RESOURCE_SEQ_NUM or
- ROUTING_SEQUENCE_ID, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, RESOURCE_SEQ_NUM or
- ASSEMBLY_ITEM_ID, ASSEMBLY_ITEM_NUMBER, ALTERNATE_ROUTING_DESIGNATOR, ORGANIZATION_ID, ORGANIZATION_CODE, OPERATION_SEQ_NUM, EFFECTIVITY_DATE, and RESOURCE_SEQ_NUM

Each imported record must have a value for the following columns.

- TRANSACTION_TYPE (Delete)
- PROCESS_FLAG (1)

Derived Data

Creating a Resource

The Bill and Routing Interface program derives or defaults most of the data required to assign resources to an operation. You can optionally include a value for any derived or defaulted column, as well as data for any of the other columns in the interface table.

The Interface program uses the same logic to derive or default column values in the BOM_OP_RESOURCES_INTERFACE table as in the BOM_OP_ROUTINGS_INTERFACE and BOM_OP_SEQUENCES_INTERFACE tables; when you populate a column in the interface table, the Bill and Routing Interface program imports the row with the data you include. However, if you do not enter data in a derived or defaulted column, the program automatically imports the row with the derived or defaulted value.

The BOM_OP_RESOURCES_INTERFACE table contains columns that let you enter a user–friendly value in the interface table and the program uses the value to derive a unique identifier. The following table lists these columns and the corresponding column that stores the derived value.

BOM_OP_RESOURCES_INTERFACE User–Friendly Column Name	Derived Column Name
ASSEMBLY_ITEM_NUMBER	ASSEMBLY_ITEM_ID
RESOURCE_CODE	RESOURCE_ID
ORGANIZATION_CODE	ORGANIZATION_ID
ACTIVITY	ACTIVITY_ID

Table 2 – 15 BOM Resources Interface, Derived Columns (Page 1 of 1)

Attention: If you enter a value for the ASSEMBLY_ITEM_NUMBER column, you must insert the system item flexfield separator between each segment of your item number. When the Bill and Routing Interface program derives the segment values for the item, it searches for this separator to indicate the end of one segment value and the start of the next segment value. See: Defining Key Flexfield Segments, *Oracle Applications Flexfields Guide*

Although the OPERATION_SEQUENCE_ID is a required column, the Bill and Routing Interface program derives a value for the column if you do not include a value. The program derives a value for the OPERATION_SEQUENCE_ID column if you can populate the following columns:

- OPERATION_SEQ_NUM
- EFFECTIVITY_DATE
- ROUTING_SEQUENCE_ID

And, the program derives a value for the ROUTING_SEQUENCE_ID column if you insert values for the following columns:

- ASSEMBLY_ITEM_ID
- ALTERNATE_ROUTING_DESIGNATOR
- ORGANIZATION_ID

Updating a Resource

The Bill and Routing Interface program does not derive or default data when updating a routing.

Deleting a Resource

The Bill and Routing Interface program does not derive or default data when deleting a routing.

Optional Data

Creating a Resource

You can populate the descriptive flexfield segment columns in the interface table to import values for the resource descriptive flexfield. You define the structure for the descriptive flexfield in the column ATTRIBUTE_CATEGORY and you can populate each segment value in the columns ATTRIBUTE1 through ATTRIBUTE15. After importing a routing, you can view your descriptive flexfield values in the Resources window.

Updating a Resource

The optional data for updating a resource is the same as creating a resource in the BOM_OP_RESOURCES_INTERFACE table.

Deleting a Resource

Once the resource is identified the entire record will be placed in a delete group. There is no optional data.

Importing Additional Routing Information

When you create routings using the Bill and Routing Interface program, you can also insert multiple routing revisions using the MTL_RTG_ITEM_REVS_INTERFACE table.

Inserting Routing Details

To import multiple routing revisions, you must insert the data into the MTL_RTG_ITEM_REVS_INTERFACE table.

See Also

Define Routing, *Oracle Bills of Material User's Guide*

Required Data

To Insert, Update or Delete data into the MTL_RTG_ITEM_REVS_INTERFACE table, you must assign a value to the following columns:

- PROCESS_REVISION
- ORGANIZATION_ID
- ORGANIZATION_CODE
- INVENTORY_ITEM_ID
- ITEM_NUMBER
- PROCESS_FLAG (1)
- TRANSACTION_TYPE

Derived Data

When you insert data into the MTL_RTG_ITEM_REVS_INTERFACE table, the Bill and Routing Interface program defaults the system date for the EFFECTIVITY_DATE column if you do not enter a value.

Optional Data

You can optionally assign values to any of the other columns in the MTL_RTG_ITEM_REVS_INTERFACE tables.

Validating Interface Table Rows

After you load the routing, routing revision, operation, and resources, the Bill and Routing Interface program validates the required data for the four interface tables. Routing validation insures that each row has an included or defaulted value for all the required columns, and verifies any interdependent relationships. The program validates the data the same way as the Define Routing form and the Define Engineering Routing form validate manually entered routings. For example, you cannot have a routing that references a common routing in a different organization.

When you import routings, the Bill and Routing Interface program validates all rows in the BOM_OP_ROUTINGS_INTERFACE, BOM_OP_SEQUENCES_INTERFACE, BOM_OP_RESOURCES_INTERFACE, and

MTL_RTG_ITEM_REVS_INTERFACE tables that have a PROCESS_FLAG set to **1** (Pending).

If the Bill and Routing Interface program cannot assign a value to a row or validate that row, the program sets the PROCESS_FLAG for the row to **3** (Assign / Validation Failed) and inserts a row in the MTL_INTERFACE_ERRORS table. To identify the error message for a failed row, the program automatically populates the TRANSACTION_ID column in the error interface table with the TRANSACTION_ID value. Each error also has a value for the MESSAGE_NAME and REQUEST_ID columns in the error interface table. The MESSAGE_NAME column corresponds to messages stored in Oracle Application Message Dictionary and the REQUEST_ID column stores the concurrent request id. If the program detects any internal database errors, the program immediately aborts and writes an error to the log file. See: Define Messages, *Oracle Application Object Library Reference Manual*

When the Bill and Routing Interface successfully assigns values and validates rows, the program inserts the rows into the BOM_OPERATIONAL_ROUTINGS, BOM_OPERATION_SEQUENCES, BOM_OPERATION_RESOURCES, and, if appropriate, the MTL_RTG_ITEM_REVISIONS tables for each validated row. If the program cannot insert the row into one of these production tables, the program sets the PROCESS_FLAG for that row to **3** (Import Failed).

After the Bill and Routing Interface program successfully creates a routing in the production tables, it sets the PROCESS_FLAG column to **7** (Import Succeeded) for the successfully processed rows. You can optionally delete all the successfully processed rows from the interface tables, after the records have been imported.

If you import a routing with multiple operations and one of the operations fails validation, the Bill and Routing Interface program imports the routing and the other validated operations. For this interface table, the program only fails the record where the error occurred. The same is true for resources and revisions.

Resolving Failed Interface Table Rows

Reviewing Failed Rows

You can review and report rows in the interface tables using SQL*Plus or any custom reports you develop. Since all rows in the interface table

have a value for PROCESS_FLAG, you can easily identify records that are successfully imported into Oracle Bills of Material and Oracle Engineering, or records that failed validation or import. You can also identify individual records by the unique value for the TRANSACTION_ID column.

Correcting Failed Rows

You can update any row from the interface tables using SQL*Plus. If you update a row to resolve invalid data, you must set the PROCESS_FLAG to **1** (Pending) for that row. If you delete a failed row and insert a replacement row, you should set the PROCESS_FLAG to **1** (Pending) for the new row. When you resubmit the Bill and Routing Interface program all rows pending validation are processed.

CHAPTER

3

Oracle Inventory Open Interfaces

This chapter contains information about the following Oracle Inventory open 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:

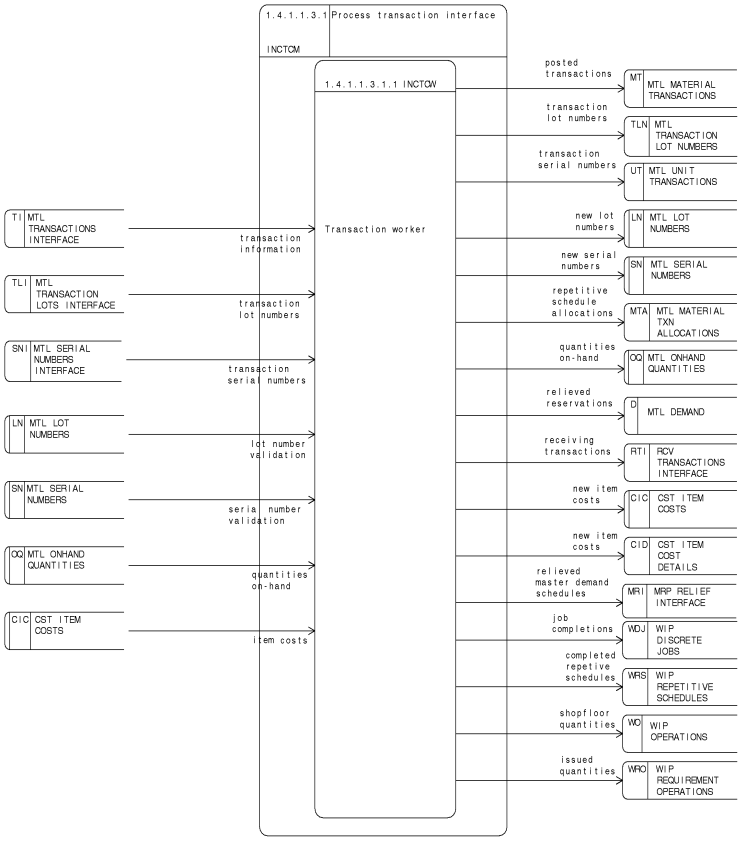
- Inventory issues and receipts (including user–defined transaction types)
- Subinventory transfers
- Direct inter–organization transfers
- Intransit shipments
- WIP component issues and returns
- WIP assembly completions and returns
- Sales order shipments
- Inventory average cost updates

This interface is also used as an integration point with Oracle Order Entry for shipment transactions. Oracle Order Entry'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 3 – 1



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.

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.

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.

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)	✓		
SOURCE_LINE_ID	Number	✓		
SOURCE_HEADER_ID	Number	✓		
PROCESS_FLAG	Number(1)	✓		
TRANSACTION_MODE	Number	✓		
LOCK_FLAG	Number(1)			✓
TRANSACTION_HEADER_ID	Number		✓	
ERROR_CODE	Varchar2(240)		✓	
ERROR_EXPLANATION	Varchar2(240)		✓	
VALIDATION_REQUIRED	Number			✓
TRANSACTION_INTERFACE_ID	Number	✓		
INVENTORY_ITEM_ID	Number	✓		
ITEM_SEGMENT1 to ITEM_SEGMENT20	Varchar2(40)	✓		
REVISION	Varchar2(3)	1		
ORGANIZATION_ID	Number	✓		
SUBINVENTORY_CODE	Varchar2(10)	2		
LOCATOR_ID	Number	3		

Table 3 – 1 Transaction Interface (Page 1 of 5)

Column Name	Type	Required	Derived	Optional
LOC_SEGMENT1toLOC_SEGMENT20	Varchar2(40)	3		
TRANSACTION_QUANTITY	Number	✓		
TRANSACTION_UOM	Varchar2(3)	✓		
PRIMARY_QUANTITY	Number		✓	
TRANSACTION_DATE	Date	✓		
ACCT_PERIOD_ID	Number		✓	
TRANSACTION_SOURCE_ID	Number	✓		
DSP_SEGMENT1toDSP_SEGMENT30	Varchar2(40)	✓		
TRANSACTION_SOURCE_NAME	Varchar2(30)	✓		
TRANSACTION_SOURCE_TYPE_ID	Number		✓	
TRANSACTION_ACTION_ID	Number		✓	
TRANSACTION_TYPE_ID	Number	✓		
REASON_ID	Number			✓
TRANSACTION_REFERENCE	Varchar2(240)			✓
TRANSACTION_COST	Number	4		
DISTRIBUTION_ACCOUNT_ID	Number	5		
DST_SEGMENT1toDST_SEGMENT30	Varchar2(25)	5		
CURRENCY_CODE	Varchar(30)			✓
CURRENCY_CONVERSION_TYPE	Varchar(30)			✓
CURRENCY_CONVERSION_RATE	Number			✓
CURRENCY_CONVERSION_DATE	Date			✓
USSGL_TRANSACTION_CODE	Varchar(30)			✓
ENCUMBRANCE_ACCOUNT	Number			✓
ENCUMBRANCE_AMOUNT	Number			✓
VENDOR_LOT_NUMBER	Varchar2(30)			✓

Table 3 – 1 Transaction Interface (Page 2 of 5)

Column Name	Type	Required	Derived	Optional
TRANSFER_SUBINVENTORY	Varchar2(10)	6		
TRANSFER_ORGANIZATION	Number	6		
TRANSFER_LOCATOR	Number	3,6		
XFER_LOC_SEGMENT1toX-FER_LOC_SE	Varchar2(40)	3,6		
SHIPMENT_NUMBER	Varchar2(30)	7		
TRANSPORTATION_COST	Number			✓
TRANSPORTATION_ACCOUNT	Number			✓
TRANSFER_COST	Number			✓
FREIGHT_CODE	Varchar2(25)			✓
CONTAINERS	Number			✓
WAYBILL_AIRBILL	Varchar2(20)			✓
EXPECTED_ARRIVAL_DATE	Date			✓
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			✓
TRX_SOURCE_LINE_ID	Number			9

Table 3 – 1 Transaction Interface (Page 3 of 5)

Column Name	Type	Required	Derived	Optional
TRX_SOURCE_DELIVERY_ID	Number			9
CUSTOMER_SHIP_ID	Number			✓
SHIPPABLE_FLAG	Varchar2(1)		✓	
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number	✓		
LAST_UPDATE_LOGIN	Number			✓
REQUEST_ID	Number			✓
PROGRAM_APPLICATION_ID	Number			✓
PROGRAM_ID	Number			✓
COST_GROUP_ID	Number	8		
PROGRAM_UPDATE_DATE	Date			✓
ATTRIBUTE_CATEGORY	Varchar2(30)			✓
ATTRIBUTE1 to ATTRIBUTE15	Varchar2(150)			✓
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

Table 3 – 1 Transaction Interface (Page 4 of 5)

Column Name	Type	Required	Derived	Optional
SCHEDULE_GROUP	Number			14
BUILD_SCHEDULE	Number			14
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

Table 3 – 1 Transaction Interface (Page 5 of 5)

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: page 3 – 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:

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

Table 3 – 2 Transaction Types and Internal IDs (Page 1 of 1)

You can identify the TRANSACTION_TYPE_ID for user–defined transactions by selecting from MTL_TRANSACTION_TYPES where TRANSACTION_TYPE_NAME is the transaction type you wish to use.

TRANSACTION_SOURCE_TYPE_ID

This column is derived from MTL_TRANSACTION_TYPES using the value you enter in TRANSACTION_TYPE_ID.

TRANSACTION_SOURCE_NAME

This column is required for user–defined transaction source types. Enter the value of the source name, such as an order number, to be displayed on all transaction reports and inquiries.

TRANSACTION_SOURCE_ID

TRANSACTION_SOURCE_ID or the corresponding flexfield segment columns (DSP_SEGMENT1 to DSP_SEGMENT30) are required for all transaction source types other than those that are user–defined. You should enter the foreign key ID that points to the context table identified by the transaction source type.

Source Type	Foreign Key Reference
Account	GL_CODE_COMBINATIONS.CODE_COMBINATION_ID
Account Alias	MTL_GENERIC_DISPOSITIONS.DISPOSITION_ID
Job or Schedule	WIP_ENTITIES.WIP_ENTITY_ID
Sales Order	MTL_SALES_ORDERS.SALES_ORDER_ID

Table 3 – 3 TRANSACTION_SOURCE_ID, Foreign Key References (Page 1 of 1)

DSP_SEGMENT1 TO DSP_SEGMENT30

You can use these flexfield segment columns instead of TRANSACTION_SOURCE_ID to enter the more user–friendly information. For example, if the interfaced transaction is for an Issue to Account transaction type, you would enter the GL Code Combination segment values in these columns instead of putting the Code GL Code Combination ID in TRANSACTION_SOURCE_ID.

TRANSACTION_ACTION_ID

This column is derived from MTL_TRANSACTION_TYPES using the value you enter in TRANSACTION_TYPE_ID.

OPERATION_SEQ_NUM

For assembly completions and returns, this value is derived. For WIP component issues and returns with routings, this value is required. For WIP routings, enter 1.

WIP_ENTITY_TYPE

For WIP component issues and returns, and WIP assembly completions and returns, enter one of the following values:

1 – Standard discrete jobs

2 – Repetitive schedules

3 – Non–standard discrete jobs

4 – Work Order–less Schedule

REASON_ID

Use this column to specify a transaction reason from the predefined list of reasons in MTL_TRANSACTION_REASONS.

TRANSACTION_REFERENCE

Use this column to enter any transaction reference information to be displayed in transaction inquiries and reports.

TRANSACTION_COST

You can use this column to specify a transaction unit cost for average cost Inventory issues and receipts. If you leave it blank, the current system unit cost is used.

DISTRIBUTION_ACCOUNT_ID

Use this column (or the flexfield segment columns) to specify the account to charge for the cost of the Inventory transaction. It is required for user–defined transactions, and derived by the Transaction Worker based on the transaction source type and source for Account Issue/Receipt and Account Alias Issue/Receipt transactions.

DST_SEGMENT1 TO DST_SEGMENT30

You can use these flexfield segment columns instead of DISTRIBUTION_ACCOUNT_ID to enter the more user–friendly information. For example, if the interfaced transaction is for an Issue to Account transaction type, you would enter the GL Code Combination segment values in these columns instead of putting the Code GL Code Combination ID in DISTRIBUTION_ACCOUNT_ID.

CURRENCY_CODE

If your transaction cost is in a different currency than the functional currency of your set of books, enter the currency code.

CURRENCY_CONVERSION_TYPE

If you enter a currency code other than the functional currency for your set of books, enter the conversion type.

CURRENCY_CONVERSION_RATE

If you enter a currency code other than the functional currency for your set of books, enter the conversion rate

CURRENCY_CONVERSION_DATE

Enter the currency conversion date for which the conversion rate is valid for the transaction.

VENDOR_LOT_NUMBER

Use this column as transaction reference information and/or to cross–reference supplier lot numbers against internal lot numbers.

TRANSFER_ORGANIZATION

This column is required for all inter–organization transfers. Enter the destination organization's internal ID.

TRANSFER_SUBINVENTORY

This column is required for subinventory transfers within the same organization and direct transfers from one organization to another. For these scenarios, enter the destination subinventory.

TRANSFER_LOCATOR

This column is required for subinventory transfers within the same organization and direct transfers from one organization to another when the item being transferred is under locator control in the destination subinventory. For these scenarios, enter the destination locator internal ID.

XFER_LOC_SEGMENT1–XFER_LOC_SEGMENT20

When a transfer locator is required, you can optionally use these columns instead of TRANSFER_LOCATOR when you want to use the user–friendly flexfield representation of the transfer locator instead of the internal ID.

SHIPMENT_NUMBER

This column is required for intransit shipments. It groups shipment lines in RCV_SHIPMENT_LINES under a parent shipment number in RCV_SHIPMENT_HEADERS.

The Transaction Worker will not process intransit transactions if a shipment header already exists in RCV_SHIPMENT_HEADERS that matches SHIPMENT_NUMBER. If you want to group shipment lines under the same header, you must ensure they are processed by the same worker. You can accomplish this using the concurrent processing mode, using the TRANSACTION_HEADER_ID to group your interface transactions, and directly calling a Transaction Worker to process that group.

NEW_AVERAGE_COST

Average cost update transactions require that either NEW_AVERAGE_COST, VALUE_CHANGE, or PERCENTAGE_CHANGE be populated, depending on the type of cost update being performed.

VALUE_CHANGE

See NEW_AVERAGE_COST.

PERCENTAGE_CHANGE

See NEW_AVERAGE_COST.

DEMAND_ID

Use this column for sales order shipment transactions to identify the exact reservation row to be relieved in MTL_DEMAND. If you do not have the DEMAND_ID information, leave this column blank, and the Transaction Processor will try to match reservations to relieve by checking MTL_DEMAND to see if there are any reservations where there is a match on:

MTL_TRANSACTIONS_INTERFACE	MTL_DEMAND
ORGANIZATION_ID	ORGANIZATION_ID

Table 3 – 4 Table Mapping: MTL_TRANSACTIONS_INTERFACE to MTL_DEMAND (Page 1 of 2)

MTL_TRANSACTIONS_INTERFACE	MTL_DEMAND
INVENTORY_ITEM_ID	INVENTORY_ITEM_ID
TRANSACTION_SOURCE_TYPE_ID	DEMAND_SOURCE_TYPE_ID
TRANSACTION_SOURCE_ID	DEMAND_SOURCE_HEADER_ID
TRANSACTION_SOURCE_LINE_ID	DEMAND_SOURCE_LINE_ID
TRANSACTION_SOURCE	DEMAND_SOURCE
DELIVERY_ID	DELIVERY_ID

Table 3 – 4 Table Mapping: MTL_TRANSACTIONS_INTERFACE to MTL_DEMAND (Page 2 of 2)

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_ID

See DEMAND_SOURCE_HEADER_ID.

DEMAND_SOURCE_DELIVERY_ID

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:

Column Name	Type	Required	Derived	Optional
TRANSACTION_INTERFACE_ID	Number	✓		
SOURCE_CODE	Varchar2(30)			✓
SOURCE_LINE_ID	Number			✓
LOT_NUMBER	Varchar2(30)	✓		
LOT_EXPIRATION_DATE	Date	1		
TRANSACTION_QUANTITY	Number	✓		
PRIMARY_QUANTITY	Number		✓	
SERIAL_TRANSACTION_TEMP_ID	Number	2		
ERROR_CODE	Varchar2(30)		✓	
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number	✓		
LAST_UPDATE_LOGIN	Number			✓
REQUEST_ID	Number			✓
PROGRAM_APPLICATION_ID	Number			✓
PROGRAM_ID	Number			✓

Table 3 – 5 Transaction Lot Numbers Interface (Page 1 of 2)

Column Name	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	Date			✓
Notes [1] If item is under lot expiration control [2]If item is under both lot and serial control				

Table 3 – 5 Transaction Lot Numbers Interface (Page 2 of 2)

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:

Column Name	Type	Required	Derived	Optional
TRANSACTION_INTERFACE_ID	Number	✓		
SOURCE_CODE	Varchar2(30)			✓
FM_SERIAL_NUMBER	Varchar2(30)	✓		
TO_SERIAL_NUMBER	Varchar2(30)			✓
SOURCE_LINE_ID	Number			✓

Table 3 – 6 Transaction Serial Numbers Interface (Page 1 of 2)

Column Name	Type	Required	Derived	Optional
VENDOR_SERIAL_NUMBER	Varchar2(30)			✓
ERROR_CODE	Varchar2(30)		✓	
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number	✓		
LAST_UPDATE_LOGIN	Number			✓
REQUEST_ID	Number			✓
PROGRAM_APPLICATION_ID	Number			✓
PROGRAM_ID	Number			✓
PROGRAM_UPDATE_DATE	Date			✓

Table 3 – 6 Transaction Serial Numbers Interface (Page 2 of 2)

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 rightmost 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.

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 Demand Interface

The Demand Interface provides all the functions you need to interface an external order entry system with Oracle Inventory and Oracle Manufacturing applications. It provides a two–way interface that lets you:

- Provide visibility to demand created in external applications for forecasting, planning, and order promising purposes. This includes the option to automatically check ATP when adding demand to verify availability.
- Reserve on–hand inventory to specific sources of demand (for example, sales orders). This includes the option to 'auto–reserve' inventory at a detailed level for warehouse picking.
- Check Available to Promise (ATP), Available to Reserve (ATR), and on–hand quantity information.
- Specify Assemble to Order Model and Option demand details for creation of and / or association with an ATO Configured Item.

The purpose of this essay is to explain how to use the Demand Interface to integrate other applications with Oracle Inventory.

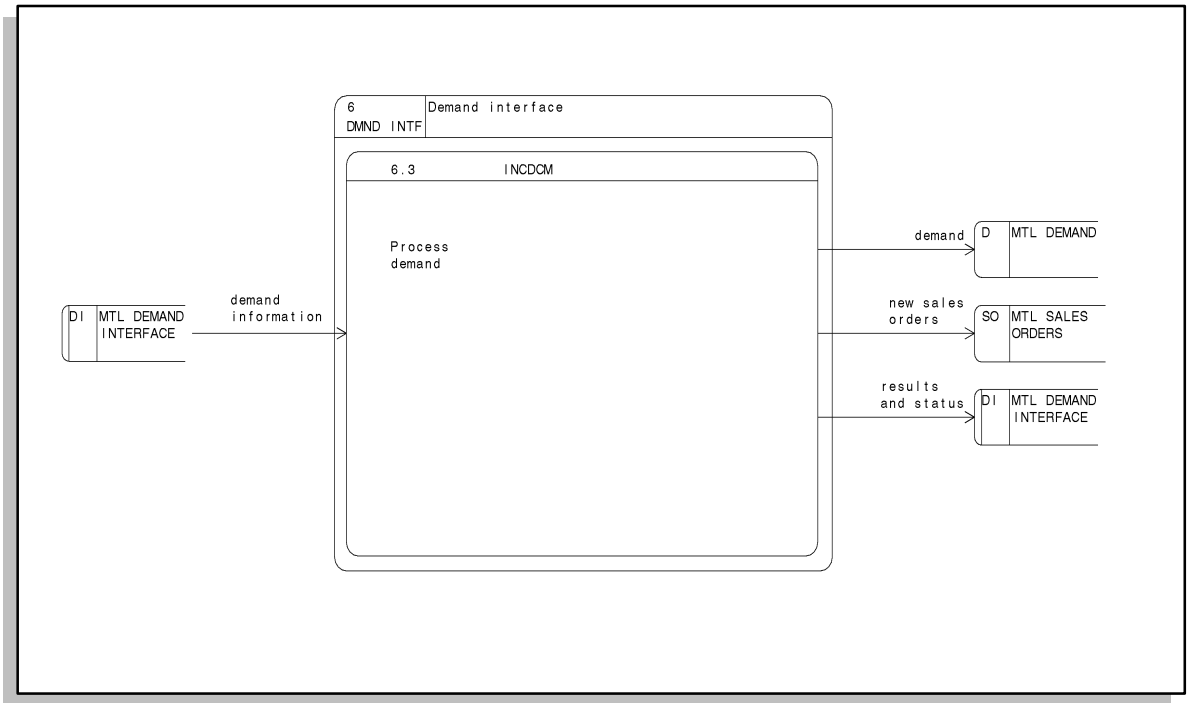
Functional Overview

The Demand Interface supports several functions or 'actions' that you can use to perform the above operations. These include:

- Demand Add
- Reservation Add
- Reservation Modify
- Demand and Reservation Add
- Update Forecast Attributes
- Pick Release
- ATS Query (request available and on–hand quantity information)
- ATS QuickPick (request available and on–hand quantity details)
- ATP Check

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

Figure 3 – 2



To use the Demand Interface, you need to insert rows into MTL_DEMAND_INTERFACE that pass all required information to execute the above functions. The rows you insert contain action codes (ACTION_CODE) that tell the Demand Manager which function(s) to execute. You group your interface transactions using a common identifier SCHEDULE_GROUP_ID. If any row in the group fails processing, the Demand Manager fails all rows. The Demand Manager processes only rows with PROCESS_FLAG = 1 (Yes) and ERROR_CODE = null.

When you are placing or updating demand and reservations, the flow of data passes through the interface table to MTL_DEMAND. In some cases, you pass a unique identifier that you can subsequently use to query results from MTL_DEMAND on rows that were created or modified by your request. If errors occur, the Demand Manager leaves

the inserted row in the interface table and updates it with an error code and explanation.

When you are using the Demand Interface to query for on–hand, available to reserve, or available to promise information, the Demand Manager takes the rows inserted into the interface table as input, and either returns the results by updating the same row, or by inserting new rows back into the interface table with a unique group identifier for querying purposes.

Transaction Processing Modes

There are two basic ways that you can interface with the Demand Manager. The first way is to simply commit rows to the interface table, leaving them to be picked up and processed by the polling Demand Manager. This mode is known as 'Background' processing.

The second way is to insert the rows into the interface table and call the Demand Manager Processor directly from a form (via user–exit) or concurrent program. This gives more direct feedback to the calling function, providing a more timely, integrated solution.

You may choose to use a combination of both methods, using the background mode (TRANSACTION_MODE = 3) to maintain demand information, and using on–line or concurrent mode (TRANSACTION_MODE= 2, 1) to check available and on–hand information. See the next section for details on how to call the Demand Processor from a form or C program.

Setting Up the Demand Interface

Setting Up the Inventory Concurrent Manager

For optimal processing in the Demand 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 Demand 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.

With this configuration, the Demand Manager must contend with other processes such as the Material Transaction Manager and Transaction Workers for the same processing queue. If you have the available resources, you can substantially reduce the time to process your

interfaced demand requests by increasing the target processes and reducing the concurrent manager sleep time using the Concurrent Managers window. This will allow the Demand Manger to run in parallel with the Transaction Manager and Transaction Workers. See: Defining Managers and their Work Shifts, *Oracle Applications System Administrator's Guide.*

Starting the Demand Manager

Once you have set up the Inventory concurrent manager, you can launch the Inventory Demand Manager in the Interface Managers window. This launches the Demand Manager and lets you specify the interval to use to poll the interface table. See: Launching Transaction Managers, *Oracle Inventory User's Guide.*

Calling the Demand Processor from a Form

To call the Demand Processor from a form, you need to first populate MTL_DEMAND_INTERFACE, and then call the Oracle Inventory user–exit GROUP_PROCESS with tokens of GROUP_ID and PARTIAL_FLAG as follows:

```
# INV GROUP_PROCESS
  GROUP_ID=group_id
  [PARTIAL_FLAG=partial_flag]
```

group_id is required and identifies which rows in MTL_DEMAND_INTERFACE to process (using SCHEDULE_GROUP_ID).

partial_flag defaults to 2 (No), but can be set to 1 (Yes) to indicate that demand rows in the group can succeed even if others fail.

Calling the Demand Processor from a C Program

To call the Demand Processor from a C program, you need to first populate MTL_DEMAND_INTERFACE and then call the GrpDemIface function with the following function prototype. This function is archived in the Oracle Inventory object library (libinv.a), which references functions defined in (libfnd.a):

```
sb4 GrpDemIface(/*_ sb4 GroupId, sb2 ProcessMode,
                    sb2  PAFlag _*/);
```

sb2 and sb4 are type definitions Signed Short and Signed Long.

GroupId identifies which rows in MTL_DEMAND_INTERFACE to process (using SCHEDULE_GROUP_ID).

ProcessMode should match the value in MTL_DEMAND_INTERFACE.TRANSACTION_MODE for the rows you are processing. You should use either mode 1 or 2. With mode 2, output is directed to a log file. This should be used only for concurrent processing. Mode 1 does not write to a log file, so you should use it when calling the Demand Processor from a user–exit.

PAFlag identifies whether partial completions of the group request are allowed. Set this flag to 1 (Yes) to indicate that demand rows in the group can succeed even if others fail, 2 (No) to indicate that all must complete successfully as a group.

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 a three segment. 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 rows into the demand interface for sales order demand, you should use the Sales Order segment values to identify the order. The Demand 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.

Demand Interface Function Descriptions

This section describes each action that you can perform using the Demand Interface. It also includes for each action a list of the required and optional inputs. Since several of the demand actions provide a corresponding output through the MTL_DEMAND_INTERFACE table, the columns where the output can be found are also listed.

All functions that you perform on the demand table require that you specify a 'demand key' which contains the complete set of information

to uniquely identify rows in MTL_DEMAND. The demand key contains the following:

- Demand source type (for example, Sales Order, Account)
- Demand source. The demand table provides several columns for you to uniquely identify your demand source, such as the order, line, and shipment details. The columns are:
  - DEMAND_SOURCE_HEADER_ID
    or
  - DEMAND_HEADER_SEGMENT1 – 30 (for sales order, account, or account alias source types)
  - DEMAND_SOURCE_NAME (for user–defined transaction source types)
  - DEMAND_SOURCE_LINE
  - DEMAND_SOURCE_DELIVERY
- Organization
- Item

Demand Add

Demand Add lets you place new demand and update existing demand for a given demand source in the demand table. In addition to the demand key, you must specify the demand quantity in any valid unit of measure for the item using LINE_ITEM_QUANTITY and LINE_ITEM_UOM. The Demand Manager will automatically convert the quantity into a quantity in the item's primary unit of measure. You must also specify the REQUIREMENT_DATE indicating when you expect the demand to be fulfilled. In an ATO scenario, you must identify whether the DEMAND_TYPE is:

1. Models
2. Option Classes
3. Options

Otherwise, for standard demand, you need to specify DEMAND_TYPE of 6 (Standard).

If the demand you are creating is specific to a subinventory, you should enter the value in SUBINVENTORY. Note that this automatically restricts any subsequent reservations for this demand to available quantities within the specified subinventory.

MRP Forecast Attributes

You *must* specify the following demand attributes used by MRP for forecast consumption and loading the master demand schedule:

- CUSTOMER_ID
- BILL_TO_SITE_USE_ID
- SHIP_TO_SITE_USE_ID

You can optionally specify the following demand attribute:

- TERRITORY_ID

In an ATO Scenario, you can update these forecast attributes for all demand under a model in a single call by entering a value in RTO_MODEL_SOURCE_LINE corresponding to the value in DEMAND_SOURCE_LINE for the parent Model.

You can also enter and update the Demand Class as another option to partition your demand. This information is used for forecast consumption and / or to check ATP if you are using an ATP Rule that partitions supply and demand by Demand Class.

ATP Attributes

When you interface demand, you can request that ATP / CTP be checked as part of the validation process before the demand is placed by setting ATP_CHECK to Y (Check Material Only), C (Check Material and Resources), or R (Check Resources Only). See: Order Entry Attribute Group, *Oracle Inventory User's Guide*, Available to Promise, *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning User's Guide*, and Capable to Promise, *Oracle Master Scheduling/MRP and Oracle Supply Chain Planning User's Guide*. If you want to place demand for several demand rows only if all rows pass the ATP check, then you should also populate the ATP_GROUP_ID with a common identifier for all rows in the group.

If the ATP checks succeed, demand will be updated and the interfaced rows will be removed from the MTL_DEMAND_INTERFACE table.

ATO/BOM Attributes

When you place demand for ATO models, options, and option classes, the Demand Manager automatically identifies them from item master attributes. In certain scenarios, the Demand Manager automatically explodes down the bill of materials to create interface rows for ATP–able components (see the ATP Check section, below).

Once the ATP check succeeds (if required) the Demand Manager explodes the model, option, and option class bills of material to populate MTL_DEMAND with demand rows for any ATP–able components. You need to populate the following columns:

- COMPONENT_SEQUENCE_ID. Unique identifier in BOM structure. For Model demand, this should correspond to BOM_BILL_OF_MATERIALS.BILL_SEQUENCE_ID. For Option or Option Class demand, this should correspond to BOM_INVENTORY_COMPONENTS.COMPONENT_ SEQUENCE_ID.
- PARENT_COMPONENT_SEQ_ID. BOM structure unique identifier of parent.
- RTO_MODEL_SOURCE_LINE. The Model's DEMAND_SOURCE_LINE for all Options, Option Classes, and Components.
- EXPLOSION_EFFECTIVITY_DATE. Date used to explode bill of materials
- CONFIG_STATUS. Status code to indicate whether the model, option, option class demand rows are complete (in other words, ready for configuration item creation). The values are:

  10 – Not ready for configuration item creation

  20 – Ready for configuration item creation

Reservation Add

Reservation Add lets you reserve available inventory for demand that has already been placed. You need to specify the demand key, as well as the new reservation balance and UOM in LINE_ITEM_RESERVATION_QTY, LINE_ITEM_UOM. Reservation Add will determine if a reservation already exists with the exact same details, and either replace the existing reservation quantity, or create a new reservation row if none already exists. The following columns can optionally be used to reserve inventory to greater detail than organization and item.

- REVISION
- LOT
- SUBINVENTORY
- LOCATOR_ID or LOC_SEGMENT1 – 20

You can reserve revision, lot, and locator details only if that quantity control level is enabled. Also, if you specify reservation details, you must make sure to specify any other details higher up in the reservation hierarchy (see above list). For example, if an item is under lot control, but not revision control, and you want to reserve material in a specific subinventory, you must specify the organization, item, lot, and subinventory (no revision).

You can have multiple reservations for each demand row, but the total reserved quantity cannot exceed the demand.

Reservation Modify

Reservation Modify performs much like Reservation Add, except you need to specify the net change from the current reservation quantity in LINE_ITEM_RESERVATION_QTY. If no reservation exists for the same organization, item, revision, subinventory, and lot, Reservation Modify will create a new row. This function requires that you specify RESERVATION_TYPE=2 for reservations of available inventory.

See the Reservation Add section above for more details.

Demand and Reservation Add

This function combines the capabilities of Demand Add and Reservation Add, letting you create demand and reserve all or part of the demand quantity in one call. If the demand row does not already exist, then a new one will be created. If the demand and/or reservation rows already exists, they will be updated to the new state.

See the Demand Add and Reservation Add sections above for more details.

Update Forecast Attributes

This function can be used to update attributes used by MRP to consume the item forecast. The attributes are CUSTOMER_ID, TERRITORY_ID, BILL_TO_SITE_USE_ID, and SHIP_TO_SITE_USE_ID (see Demand Add for details). In an ATO scenario, you automatically update the attributes for exploded demand rows under a parent model, option, or option class. You can also update the attributes for all demand under a model in one call by entering a value in RTO_MODEL_SOURCE_LINE corresponding to the value in DEMAND_SOURCE_LINE for the parent Model.

Pick Release

This function takes existing demand and attempts to reserve inventory to the item's lowest inventory control level. It uses the Picking Rule defined for an item to determine which order to search for available inventory to reserve. Any existing reservations that do not completely specify all control options are replaced with 'detailed' reservations. See: Scheduling an Order or Order Line, *Oracle Order Entry/Shipping User's Guide* and Overview of Delivery–based Pick Release, *Oracle Order Entry/Shipping User's Guide*.

You need to specify what you want pick released using the demand key, and how much quantity to reserve using LINE_ITEM_QUANTITY and LINE_ITEM_UOM. You can set the PARTIALS_ALLOWED_FLAG to 1 (Yes) to indicate that Pick Release should reserve as much as possible even if the total cannot be satisfied. If this flag is not set and the entire quantity cannot be reserved, the request fails.

You need to also specify a unique identifier in AUTODETAIL_GROUP_ID. Pick Release passes this value through to MTL_DEMAND so you can query up reservation details created by your request.

If you set SINGLE_LOT_FLAG to 1 (Yes), your request will succeed only if the entire requested quantity can be fulfilled with a single lot. Pick Release searches for a complete lot quantity using the Picking Rule criteria. For items under lot expiration control, you can optionally populate LOT_EXPIRATION_CUTOFF_DATE to eliminate lots from consideration whose expiration date is earlier from inclusion in the results.

Pick Release returns the following results in the interface table:

- AUTODETAILED_QUANTITY. The total quantity successfully reserved. The column is left null if partials are not allowed and the full quantity was not able to be reserved.
- AVAILABLE_TO_RESERVE – The total quantity available to reserve regardless of whether the request succeeded
- QUANTITY_ON_HAND

ATS Query (Single–Row Availability Check)

You can use this function to request information about the availability of inventory to reserve or transact. You must specify the demand key, organization, item, and any optional item control details for which you

want information. The Demand Manager returns the requested information in QUANTITY_ON_HAND and AVAILABLE_TO_RESERVE.

For items under lot expiration control, you can optionally populate LOT_EXPIRATION_CUTOFF_DATE to eliminate lots from consideration whose expiration date is earlier from inclusion in the results.

ATS Quick Pick (Multi–Row Detail Availability Check)

This function lets you request availability information much like ATS Query. The primary difference is that you specify at what control level you want the available quantity results to be returned. For example, you can request the availability of all lots with on–hand quantity within a given organization. To accomplish this, you need to pass the organization and item in question, and specify a control level of Lot in RESERVE_LEVEL. The valid control levels are:

2 – Revision

3 – Lot

5 – Subinventory

6 – Locator

Instead of updating the original interface row, new rows are inserted into MTL_DEMAND_INTERFACE with the same SCHEDULE_GROUP_ID from the 'input row'. The results are populated in QUANTITY_ON_HAND and AVAILABLE_TO_RESERVE along with the corresponding control details (for example, revision, lot).

As with ATS Query, for items under lot expiration control, you can optionally populate LOT_EXPIRATION_CUTOFF_DATE to eliminate lots from consideration whose expiration date is earlier from inclusion in the results.

ATP Check

This function lets you check ATP for a given item within an organization. You must define the whole demand key, as well as the requested quantity in LINE_ITEM_QUANTITY (if you just want the current ATP information, you can specify a requested quantity of zero). All other inputs listed below are optional.

- LINE_ITEM_QUANTITY, LINE_ITEM_UOM. The quantity requested. If quantity entered is greater than zero, ATP Check will return the earliest date this quantity is available. If the unit of measure is not specified, the item's primary UOM is used as a default.
- REQUIREMENT_DATE. The date for which the available quantity is requested. If entered, ATP Check will return the available to promise quantity on that date.
- ATP_RULE_ID. If you populate this column, ATP Check will use it to override the default ATP rule on the Item Master. It must be a valid foreign key reference to a rule defined in MTL_ATP_RULES.
- ATP_CALENDAR_ORGANIZATION_ID. If specified, ATP Check uses the workday calendar for this organization. Otherwise, the workday calendar for the organization against which the ATP check is being performed is used.
- ATP_CHECK. This column overrides the Item Master default. Set it to Y (Check Material Only), C (Check Material and Resources), or R (Check Resources Only). This override should be used carefully since inconsistent use when placing demand could invalidate demand that was previously 'allocated' using ATP Check.
- ATP_COMPONENTS_FLAG. This column overrides the Item Master default. Set it to 1 to force explosion of an items bill of materials to include ATP–able components. If you leave this flag null, ATP Check will still explode to find ATP–able components if MTL_SYSTEM_ITEMS.ATP_COMPONENTS_FLAG = 1 (Yes) and BOM_INVENTORY_COMPONENTS.CHECK_ATP = 1 (Yes). As with ATP_CHECK, you should be careful in your use of this flag since inconsistent use while placing demand could invalidate demand that was previously 'allocated' using ATP Check.
- LATEST_ACCEPTABLE_DATE. This limits the date ATP Check will return as the GROUP_ATP_DATE.
- DEMAND_CLASS. If the ATP Rule in use is defined for Demand Class ATP, this column identifies the subset of supply and demand on which to base the query.
- ATP_LEAD_TIME. The number of days earlier than the ATP date that the item must be available. In an ATO scenario, you need to specify the assembly lead times for option classes and options to get an accurate representation of when the

subassemblies and components will be required. This is also populated by the demand manager when it explodes to find demand for ATP–able components.

- DEPARTMENT_ID, RESOURCE_ID. The department and resource are derived from the item and its CTP routing if:
  - the INV:Capable to Promise profile option is set to Enable Product Family ATP and CTP
  - And if MTL_SYSTEM_ITEM.ATP_FLAG is set to R (Resource Only) or C (Material and Resource)

ATP Check returns multiple results by updating the same interface row. The output columns are:

- REQUEST_DATE_ATP_QUANTITY. The available quantity on the request date regardless of whether the quantity is satisfied.
- EARLIEST_ATP_DATE. The earliest date that can satisfy the requested quantity regardless of the request date.
- EARLIEST_ATP_DATE_QUANTITY. The maximum quantity available on the earliest ATP date.
- REQUEST_ATP_DATE. The first date on or after the required date where the available quantity is enough to satisfy the requested quantity.
- REQUEST_ATP_DATE_QUANTITY. The maximum available quantity on the requested ATP date.
- INFINITE_TIME_FENCE_DATE. The date offset from the current date by the Infinite Supply Time Fence.

In addition to checking ATP for items one at a time, you can use the Demand Interface to perform a Group ATP check. You pass multiple rows containing different item / organization combinations, and populate ATP_GROUP_ID with a common group identifier. ATP Check not only returns all information for each row as described above, but it also determines the earliest date, if any, before the requirement date that all requests can be satisfied. This result is returned in GROUP_AVAILABLE_DATE.

Some inputs are specifically for Supply Chain ATP. Supply Chain ATP lets you check ATP within all possible sources of supply for your order line, ship set, or configuration. In addition to the columns you specify for checking ATP, you must enter the customer and customer ship–to site in CUSTOMER_ID and SHIP_TO_SITE_ID. Supply Chain ATP automatically populates VENDOR_ID and VENDOR_SITE_ID if the organization is a supplier organization.

SHIP_METHOD and INTRANSIT_LEAD_TIME are derived based on the definition of transit lead times in Oracle Inventory. They represent the intransit lead time for a shipping method between the supply organization location and the customer location. In Supply Chain APT, the group available date is the expected receipt date minus the intransit lead time.

Inserting into the Demand Interface Table

MTL_DEMAND_INTERFACE Table Description – Inputs

The following graphic describes the input columns to the MTL_DEMAND_INTERFACE table cross–referenced against the Demand Interface functions. Required columns are identified with R, optional columns with O, and derived columns with D.

MTL_DEMAND_INTERFACE Column Name	Type	Demand Add	Reserva –tion Add	Reserva –tion Modify	Demand /Reserve Add	Update Attr	Pick Release	ATS Query	ATS Quick Pick	ATP Check
SCHEDULE_GROUP_ID	Number	R	R	R	R	R	R	R	R	R
TRANSACTION_PROCESS_ORDER	Number	O	O	O	O	O	O	O	O	O
DEMAND_SOURCE_TYPE	Number	R	R	R	R	R	R	R	R	R
DEMAND_SOURCE_HEADER_ID or DEMAND_HEADER_SEGMENT1 – 30 or DEMAND_SOURCE_NAME	Varchar2(40)	R	R	R	R	R	R	R	R	R
DEMAND_SOURCE_LINE	Varchar2(30)	R	R	R	R	R	R	R	R	R
DEMAND_SOURCE_DELIVERY	Varchar2(30)	R	R	R	R	R	R	R	R	R
USER_LINE_NUM	Varchar2(30)	O			O					
USER_DELIVERY	Varchar2(30)	O			O					
EXTERNAL_SOURCE_CODE	Varchar2(30)	O	O	O	O	O	O	O	O	O
EXTERNAL_SOURCE_LINE_ID	Number	O	O	O	O	O	O	O	O	O
ACTION_CODE	Number	R	R	R	R	R	R	R	R	R
PROCESS_FLAG	Number	O	O	O	O	O	O	O	O	O
LOCK_FLAG	Number									
VALIDATE_ROWS	Number	O	O	O	O	O	O	O	O	O

Table 3 – 7 Demand Interface – Input Columns (Page 1 of 3)

MTL_DEMAND_INTERFACE Column Name	Type	Demand Add	Reserva –tion Add	Reserva –tion Modify	Demand /Reserve Add	Update Attr	Pick Release	ATS Query	ATS Quick Pick	ATP Check
TRANSACTION_MODE	Number	R	R	R	R	R	R	R	R	R
ORGANIZATION_ID or ORGANIZATION_NAME	Number Varchar2(60)	R	R	R	R	R	R	R	R	R
INVENTORY_ITEM_ID or ITEM_SEGMENT1 – 20	Number Varchar2(40)	R	R	R	R	R	R	R	R	R
REVISION	Varchar2(3)		O	O	O			O	O	
LOT_NUMBER	Varchar2(30)		O	O	O			O	O	
SUBINVENTORY	Varchar2(10)	O	O	O	O			O	O	
LOCATOR_ID or LOC_SEGMENT1 – 20	Number Varchar2(40)		O	O	O			O	O	
LINE_ITEM_QUANTITY	Number	R			R		R			
LINE_ITEM_RESERVATION_QTY	Number		R	R	R		R			
LINE_ITEM_UNIT_OF_MEASURE or LINE_ITEM_UOM	Varchar2(25) Varchar2(3)	O	O	O	O		O	O	O	O
PRIMARY_UOM_QUANTITY	Number	D			D		D	D	D	
PRIMARY_UOM	Varchar2(3)	D	D	D	D		D	D	D	
RESERVATION_QUANTITY	Number		D	D	D					
REQUIREMENT_DATE	Date	0			0					0
DEMAND_TYPE	Number	R			R					
RESERVATION_TYPE	Number			R						
AUTODETAIL_GROUP_ID	Number						R			
SINGLE_LOT_FLAG	Number						O			
RESERVE_LEVEL	Number						O			
DEMAND_CLASS	Varchar2(30)	O			O					O
CUSTOMER_ID	Number	O			O	R				O
TERRITORY_ID	Number	O			O	R				
BILL_TO_SITE_USE_ID	Number	O			O	R				
SHIP_TO_SITE_USE_ID	Number	O			O	R				O
LOT_EXPIRATION_CUTOFF_DATE	Date						O	O	O	
PARTIALS_ALLOWED_FLAG	Number						O			

Table 3 – 7 Demand Interface – Input Columns (Page 2 of 3)

MTL_DEMAND_INTERFACE Column Name	Type	Demand Add	Reserva –tion Add	Reserva –tion Modify	Demand /Reserve Add	Update Attr	Pick Release	ATS Query	ATS Quick Pick	ATP Check
ATP_CHECK	Number	O			O					O
ATP_GROUP_ID	Number	O			O					O
ATP_RULE_ID	Number	O			O					O
ATP_COMPONENTS_FLAG	Number	O			O					O
LATEST_ACCEPTABLE_DATE	Date	O			O					O
ATP_CALENDAR_ORGANIZATION _ID	Number	O			O					O
ATP_LEAD_TIME	Number	O			O					O
BOM_LEVEL	Number									
EXPLOSION_GROUP_ID	Number									
EXPLOSION_EFFECTIVITY_DATE	Date	O			O					
COMPONENT_SEQUENCE_ID	Number	O			O					
PARENT_COMPONENT_SEQ_ID	Number	O			O					
CONFIG_STATUS	Number	O			O					
RTO_MODEL_SOURCE_LINE	Varchar2(30)	O			O	O				
ATTRIBUTE_CATEGORY	Varchar2(30)	O			O					
ATTRIBUTE1 – 15	Varchar2(240)	O			O					
VENDOR_ID	Number									D
VENDOR_SITE_ID	Number									D
INTRANSIT_LEAD_TIME	Number									D
SHIP_METHOD	Varchar2(30)									D
RESOURCE_ID	Number									D
DEPARTMENT_ID	Number									D

Table 3 – 7 Demand Interface – Input Columns (Page 3 of 3)

MTL_DEMAND_INTERFACE Table Description – Outputs

The following graphic describes the output columns from the MTL_DEMAND_INTERFACE table cross–referenced against the Demand Interface functions. Several functions by their nature are only entering / updating information in MTL_DEMAND, and therefore will get an output only if an error occurs. Others, such as ATP Check, expect outputs as a result of successful completion.

MTL_DEMAND_INTERFACE Column Name	Type	Demand Add	Reserva –tion Add	Reserva –tion Modify	Demand /Reserve Add	Update Attr	Pick Release	ATS Query	ATS Quick Pick	ATP Check
ERROR_CODE	Number	✓	✓	✓	✓	✓	✓	✓	✓	✓
INFINITE_TIME_FENCE_DATE	Date	✓			✓					✓
REQUEST_DATE_ATP_QUANTITY	Number	✓			✓					✓
REQUEST_ATP_DATE	Date	✓			✓					✓
REQUEST_ATP_DATE_QUANTITY	Number	✓			✓					✓
EARLIEST_ATP_DATE	Date	✓			✓					✓
EARLIEST_ATP_DATE_QUANTITY	Number	✓			✓					✓
GROUP_AVAILABLE_DATE	Date	✓			✓					✓
AUTODETAILED_QUANTITY	Number						✓			
AVAILABLE_TO_RESERVE	Number						✓	✓	✓	
QUANTITY_ON_HAND	Number						✓	✓	✓	

Table 3 – 8 Demand Interface Table – Output Columns (Page 1 of 1)

DEMAND_SOURCE_TYPE

This column identifies the origination of the demand or reservation. The valid values for predefined source types are:

2 – Sales Order

3 – Account

6 – Account Alias

User–defined source types are defined in MTL_TXN_SOURCE_TYPES. This column is a foreign key reference to TRANSACTION_SOURCE_TYPE_ID.

DEMAND_SOURCE_HEADER_ID

DEMAND_SOURCE_HEADER_ID or the corresponding flexfield segment columns (DEMAND_HEADER_SEGMENT1 –30) 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 DEMAND_SOURCE_TYPE.

Source Type	Foreign Key Reference
Account	GL_CODE_COMBINATIONS.CODE_COMBINATION_ID
Account Alias	MTL_GENERIC_DISPOSITIONS.DISPOSITION_ID
Sales Order	MTL_SALES_ORDERS.SALES_ORDER_ID

Table 3 – 9 DEMAND_SOURCE_HEADER_ID, Foreign Key References (Page 1 of 1)

DEMAND_HEADER_SEGMENT1 – 30

You can use these flexfield segment columns instead of DEMAND_SOURCE_HEADER_ID for predefined source types (in other words, Account, Account Alias, or Sales Order). For example, if you were creating a reservation to an account, you would enter the segment values of the GL Code Combination in these columns instead of using the internal identifier.

DEMAND_SOURCE_NAME

This column is required for user–defined source types. Enter the value of the source name, such as an order number, to be displayed on all reports and inquiries related to demand.

DEMAND_SOURCE_LINE

This column is one of three 'levels' of the demand key that help you uniquely identify your demand row. The Demand Interface takes all three levels (header, line, delivery) as a whole, and does not generally distinguish between them.

DEMAND_SOURCE_DELIVERY

This column is one of three 'levels' of the demand key that help you uniquely identify your demand row. The Demand Interface takes all three levels (header, line, delivery) as a whole, and does generally distinguish between them.

USER_LINE_NUM

This column is an optional field that can be used to identify specific demand details in WIP and MRP inquiries. For example, you if you are placing demand for Order # 100, Line #2, you could populate this field

with a value of 2. The DEMAND_SOURCE_LINE could still be a unique internal identifier, but the displayed value would be more meaningful to the user.

USER_DELIVERY

You can populate this column much the same way as you use USER_LINE_NUM. It is not, however, displayed on any standard inquiries or reports.

EXTERNAL_SOURCE_CODE

You can use this column as an external system reference. The values entered are transferred directly to the transaction history table.

EXTERNAL_SOURCE_LINE_ID

You can use this column as an external system reference. The values entered are transferred directly to the transaction history table.

ACTION_CODE

Enter the code for the action to be performed on each row. The valid codes are:

Code	Action
100	ATP Check
110	Demand Add
130	Reservation Add
140	Reservation Modify
150	Demand and Reservation Add
410	Update Forecast Attributes
420	Pick Release
610	ATS Query
620	ATS Quick Pick

Table 3 – 10 Action Codes (Page 1 of 1)

Validation

Oracle Inventory lets you choose the level of validation you want performed against interfaced demand 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.

For more information on this subject, consult the details for the column ERROR_CODE in the MTL_DEMAND_INTERFACE interface table. You can find this information in the *Oracle Inventory Technical Reference Manual.*

Resolving Failed Demand Interface Rows

Viewing Failed Transactions

You can view both pending and failed demand rows in the MTL_DEMAND_INTERFACE table using the Pending Transactions window. If your requests 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 Demand 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 request. 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 requests, or by directly updating the PROCESS_FLAG and LOCK_FLAG values via SQL*PLUS.

If, however, you need to make changes to the source information because of invalid data, 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

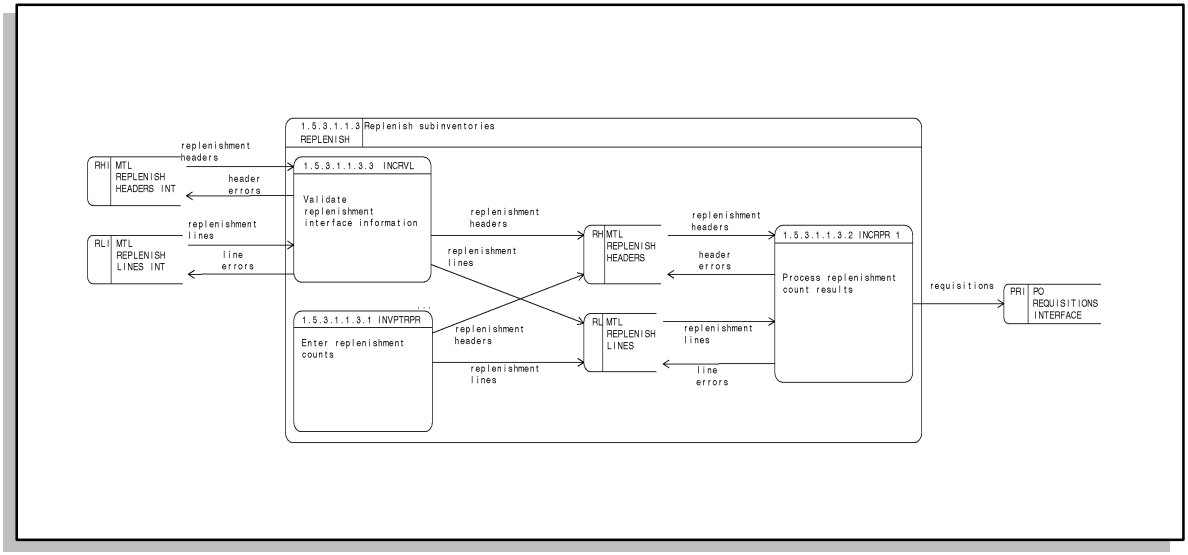
Oracle Inventory provides an open interface for you to easily load replenishment requests from external systems such as a bar–code application. Such requests may be in the form of stock–take counts or requisition requests for subinventories in which you do not track quantities.

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 3 – 3



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:

Column Name	Type	Required	Derived	Optional
REPLENISHMENT_HEADER_ID	Number	✓		
REPLENISHMENT_COUNT_NAME	Varchar2(10)	✓		
COUNT_DATE	Date	✓		
LAST_UPDATE_DATE	Date	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number	✓		
LAST_UPDATE_LOGIN	Number			✓
LAST_UPDATED_BY	Number	✓		
ORGANIZATION_ID	Number	✓		
ORGANIZATION_CODE	Varchar2(3)	✓		

Table 3 – 11 Oracle Inventory Replenishment Headers Interface (Page 1 of 2)

Column Name	Type	Required	Derived	Optional
SUBINVENTORY_CODE	Varchar2(10)	✓		
SUPPLY_CUTOFF_DATE	Date			✓
PROCESS_STATUS	Number	✓		
PROCESS_MODE	Number	✓		
ERROR_FLAG	Number		✓	
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
DELIVERY_LOCATION_ID	Number			✓
DELIVERY_LOCATION_CODE	Varchar2(20)			✓

Table 3 – 11 Oracle Inventory Replenishment Headers Interface (Page 2 of 2)

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:

Column Name	Type	Required	Derived	Optional
REPLENISHMENT_HEADER_ID	Number	✓		
REPLENISHMENT_LINE_ID	Number	✓		
ORGANIZATION_ID	Number			✓
LAST_UPDATE_DATE	Date	✓		

Table 3 – 12 Oracle Inventory Replenishment Lines Interface (Page 1 of 2)

Column Name	Type	Required	Derived	Optional
CREATION_DATE	Date	✓		
CREATED_BY	Number	✓		
LAST_UPDATE_LOGIN	Number	✓		
LAST_UPDATED_BY	Number	✓		
INVENTORY_ITEM_ID	Number	✓		
SEGMENT {1–20}	Varchar2(40)	✓		
COUNT_TYPE_CODE	Number	✓		
COUNT_QUANTITY	Number	✓		
REFERENCE	Varchar2(240)			✓
ERROR_FLAG	Number		✓	
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
COUNT_UNIT_OF_MEASURE	Varchar2(25)	✓		
COUNT_UOM_CODE	Varchar2(3)	✓		

Table 3 – 12 Oracle Inventory Replenishment Lines Interface (Page 2 of 2)

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*.

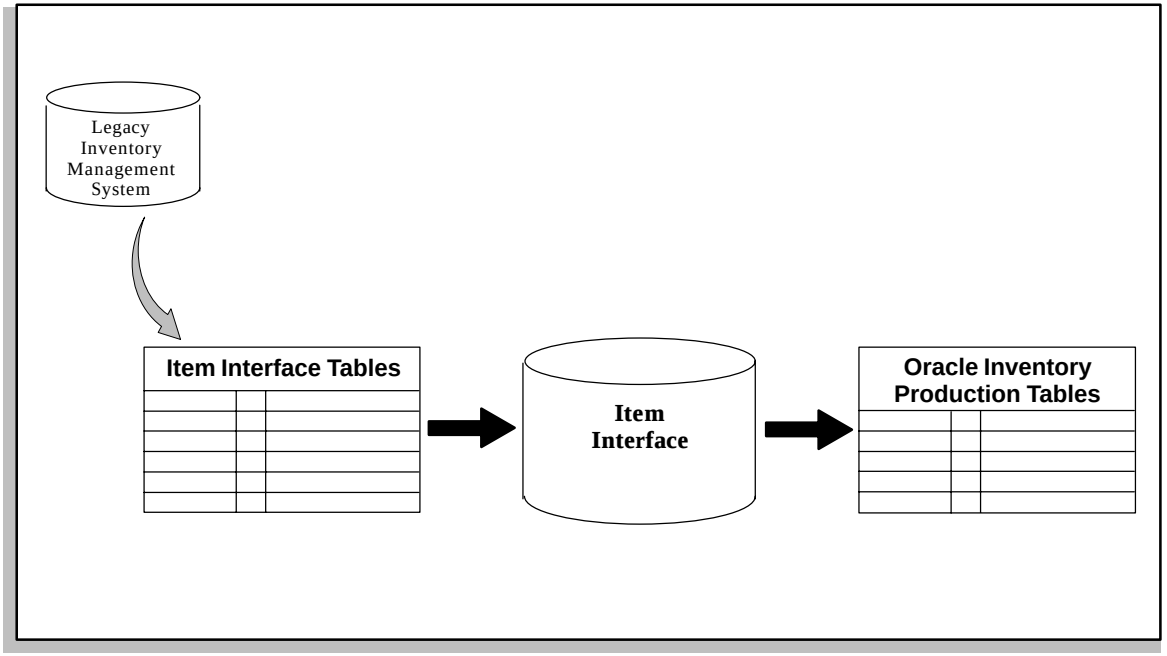
The purpose of this essay is to explain how to use the Item Interface.

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.

Figure 3 – 4



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.

MTL_SYSTEM_ITEMS_INTERFACE (Partial List of Columns) Column Name	Type	Required	Derived	Optional
ITEM_NUMBER	Varchar2(81)	*conditionally*		

Table 3 – 13 Partial List of Columns, Oracle Inventory Item Interface (Page 1 of 2)

MTL_SYSTEM_ITEMS_INTERFACE (Partial List of Columns) Column Name	Type	Required	Derived	Optional
DESCRIPTION	Varchar2(240)	*conditionally*		
MATERIAL_COST	Number			✓
MATERIAL_OVERHEAD_RATE	Number			✓
MATERIAL_OVERHEAD_SUB_ELEM	Varchar2(50)			✓
MATERIAL_OVERHEAD_SUB_ELEM_ID	Number			✓
MATERIAL_SUB_ELEM	Varchar2(50)			✓
MATERIAL_SUB_ELEM_ID	Number			✓
ORGANIZATION_CODE	Varchar2(3)	*conditionally*		
PROCESS_FLAG	Number	✓		
REVISION	Varchar2(3)			✓
TRANSACTION_ID	Number		✓	
TRANSACTION_TYPE	Varchar2(5)	✓		
SET_PROCESS_ID	Number	✓		

Table 3 – 13 Partial List of Columns, Oracle Inventory Item Interface (Page 2 of 2)

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.

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

Table 3 – 14 Meaning of PROCESS_FLAG Values (Page 1 of 1)

A full list of values for the PROCESS_FLAG is in Table 3 – 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 3 – 15.

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
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

Table 3 – 15 Column Defaults in the Item Interface (Page 1 of 4)

MTL_SYSTEM_ITEMS_INTERFACE (Partial List of Columns) Column Name	Default Value	Value Displayed in Window
LOT_CONTROL_CODE	1	No lot control
SHELF_LIFE_CODE	1	No shelf life control
SERIAL_NUMBER_CONTROL_CODE	1	No serial number control
RESTRICT_SUBINVENTORIES_CODE	2	Subinventories not restricted to predefined list
RESTRICT_LOCATORS_CODE	2	Locators not restricted to predefined list
LOCATION_CONTROL_CODE	1	No locator control
PLANNING_TIME__FENCE_CODE	4	User–defined time fence
PLANNING_TIME__FENCE_DAYS	1	1
BOM_ITEM_TYPE	4	Standard
PICK_COMPONENTS_FLAG	N	No
REPLENISH_TO_ORDER_FLAG	N	No
ATP_COMPONENTS_FLAG	N	No
ATP_FLAG	N	No
PRIMARY_UNIT_OF_MEASURE	*from profile* INV: Default Primary Unit of Measure	*from* Personal Profile Values
ALLOWED_UNITS_LOOKUP_CODE	3	Both standard and item specific
COST_OF_SALES_ACCOUNT	*from* MTL_PARAMETERS. COST_OF_SALES_ACCOUNT	*from* Organization Parameters
SALES_ACCOUNT_DSP	*from* MTL_PARAMETERS.SALES_ACCOUNT	*from* Organization Parameters
ENCUMBRANCE_ACCOUNT	*from* MTL_PARAMETERS.ENCUM-BRANCE_ACCOUNT	*from* Organization Parameters
EXPENSE_ACCOUNT	*from* MTL_PARAMETERS.EXPENSE_AC-COUNT	*from* Organization Parameters
LIST_PRICE_PER_UNIT	0	0

Table 3 – 15 Column Defaults in the Item Interface (Page 2 of 4)

MTL_SYSTEM_ITEMS_INTERFACE (Partial List of Columns) Column Name	Default Value	Value Displayed in Window
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
MRP__SAFETY_STOCK_CODE	1	Non–MRP planned
TAXABLE_FLAG	Y	*from* Purchasing Options, otherwise No
MATERIAL_BILLABLEFLAG	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

Table 3 – 15 Column Defaults in the Item Interface (Page 3 of 4)

MTL_SYSTEM_ITEMS_INTERFACE (Partial List of Columns) Column Name	Default Value	Value Displayed in Window
INVOICEABLE_ITEM_FLAG	N	No
INVOICE_ENABLED_FLAG	N	No
MUST_USE_APPROVED_VENDOR_FLAG	N	No
OUTSIDE_OPERATION_FLAG	N	No
COSTING_ENABLED_FLAG	N	No
CYCLE_COUNT_ENABLED_FLAG	N	No
AUTO_CREATED_CONFIG_FLAG	N	No
MRP_PLANNING_CODE	6	Not planned
CONTAINER_ITEM_FLAG	N	No
VEHICLE_ITEM_FLAG	N	No
END_ASSEMBLY_PEGGING_FLAG	N	None
SERVICE_DURATION	0	0
SET_PROCESS_ID	0	

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.

Table 3 – 15 Column Defaults in the Item Interface (Page 4 of 4)

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 3 – 15 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 3 – 16. 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.

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

Table 3 – 16 Oracle Inventory Item Details Interface Page 1 of 1)

Note: Although there are many other tables in Oracle Inventory whose names may imply use, the tables listed in Table 3 – 16 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 3 – 17). 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 3 – 14.

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

MTL_ITEM_REVISIONS Column Name	MTL_ITEM_REVISIONS_INTERFACE Column Source
INVENTORY_ITEM_ID	ITEM_NUMBER
ORGANIZATION_ID	ORGANIZATION_ID or ORGANIZATION_CODE
REVISION	REVISION
CHANGE_NOTICE	CHANGE_NOTICE
ECN_INITIATION_DATE	ECN_INITIATION_DATE
IMPLEMENTATION_DATE	IMPLEMENTATION_DATE
IMPLEMENTED_SERIAL_NUMBER	IMPLEMENTED_SERIAL_NUMBER
EFFECTIVITY_DATE	EFFECTIVITY_DATE
ATTRIBUTE_CATEGORY	ATTRIBUTE_CATEGORY
ATTRIBUTE*n*	ATTRIBUTE*n*
REVISED_ITEM_SEQUENCE_ID	REVISED_ITEM_SEQUENCE_ID
DESCRIPTION	DESCRIPTION

Table 3 – 17 Column–Mappings from Revision Interface Table to Oracle Inventory (Page 1 of 1)

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.

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	UpdatingMSII[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

Table 3 – 18 Oracle Inventory Item Details Interface (Page 1 of 1)

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.

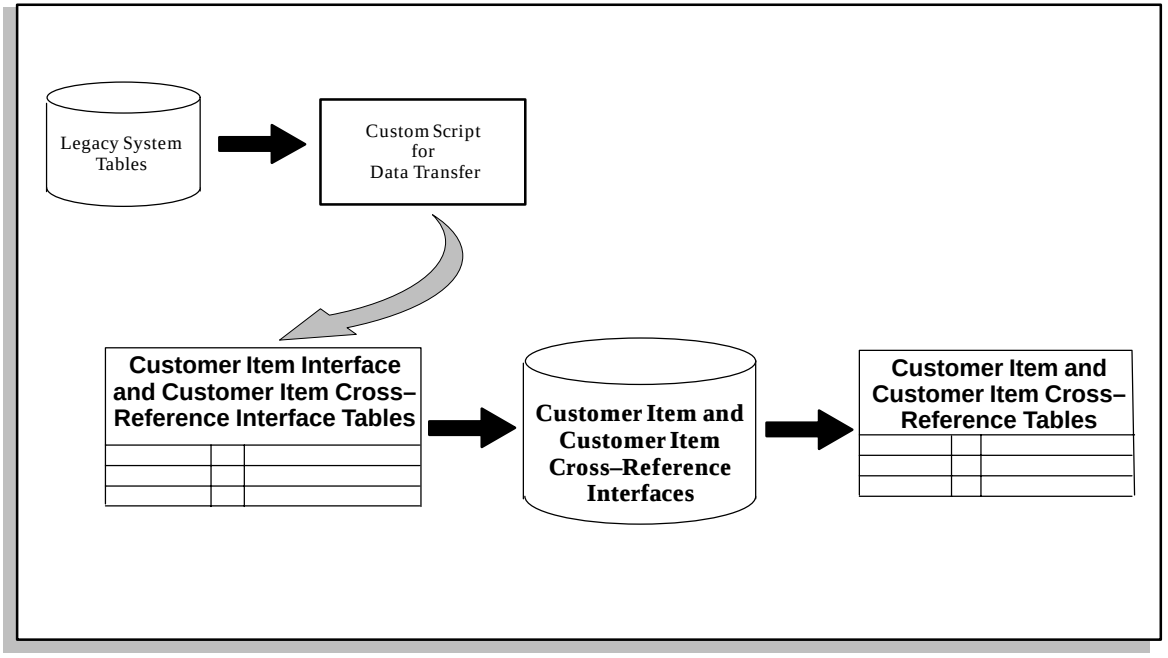
Attention: The Customer Item Interface must be run successfully before the Customer Item Cross–Reference Interface. This is to ensure that a customer item has been defined before an attempt is made to cross–reference it with an Oracle Inventory item. The Customer Item Cross–Reference Interface errors 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.

Figure 3 – 5



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: Table Administration and Audit Trail: page 3 – 97.

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

Field Name	Type	Required
PROCESS_FLAG	Varchar2(1)	✓
PROCESS_MODE	Number	✓
LAST_UPDATED_BY	Number	✓
LAST_UPDATE_DATE	Date	✓
LAST_UPDATE_LOGIN	Number	
CREATED_BY	Number	✓
CREATION_DATE	Date	✓
REQUEST_ID	Number	
PROGRAM_APPLICATION_ID	Number	
PROGRAM_ID	Number	
PROGRAM_UPDATE_DATE	Date	
TRANSACTION_TYPE	Varchar2(6)	✓
CUSTOMER_NAME	Varchar2(50)	Conditionally
CUSTOMER_NUMBER	Varchar2(30)	Conditionally
CUSTOMER_ID	Number	Conditionally
CUSTOMER_CATEGORY_CODE	Varchar2(30)	Conditionally
CUSTOMER_CATEGORY	Varchar2(80)	Conditionally
ADDRESS1	Varchar2(240)	Conditionally
ADDRESS2	Varchar2(240)	Conditionally
ADDRESS3	Varchar2(240)	Conditionally
ADDRESS4	Varchar2(240)	Conditionally
CITY	Varchar2(50)	Conditionally
STATE	Varchar2(50)	Conditionally
COUNTY	Varchar2(50)	Conditionally
COUNTRY	Varchar2(50)	Conditionally

Table 3 – 19 List of Columns, Customer Item Interface (Page 1 of 3)

Field Name	Type	Required
POSTAL_CODE	Varchar2(30)	Conditionally
ADDRESS_ID	Number	Conditionally
CUSTOMER_ITEM_NUMBER	Varchar2(50)	✓
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)	✓
ATTRIBUTE_CATEGORY	Varchar2(30)	
ATTRIBUTEn	Varchar2(150)	
DEMAND_TOLERANCE_POSITIVE	Number	
DEMAND_TOLERANCE_NEGATIVE	Number	

Table 3 – 19 List of Columns, Customer Item Interface (Page 2 of 3)

Field Name	Type	Required
ERROR_CODE	Varchar2(9)	
ERROR_EXPLANATION	Varchar2(2000)	

Table 3 – 19 List of Columns, Customer Item Interface (Page 3 of 3)

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: page 3 – 85.

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: page 3 – 97.

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.

Field Name	Type	Required	Derived	Optional
PROCESS_FLAG	Varchar2(1)	✓		
PROCESS_MODE	Number	✓		
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number(15)	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number(15)	✓		
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)	✓		
CUSTOMER_NAME	Varchar2(50)	Conditionally		
CUSTOMER_NUMBER	Varchar2(30)	Conditionally		
CUSTOMER_ID	Number	Conditionally		
CUSTOMER_CATEGORY_CODE	Varchar2(30)	Conditionally		
CUSTOMER_CATEGORY	Varchar2(80)	Conditionally		
ADDRESS1	Varchar2(240)	Conditionally		
ADDRESS2	Varchar2(240)	Conditionally		
ADDRESS3	Varchar2(240)	Conditionally		
ADDRESS4	Varchar2(240)	Conditionally		
CITY	Varchar2(50)	Conditionally		
STATE	Varchar2(50)	Conditionally		
COUNTY	Varchar2(50)	Conditionally		

Table 3 – 20 List of Columns, Customer Item Cross–Reference Interface (Page 1 of 2)

Field Name	Type	Required	Derived	Optional
COUNTRY	Varchar2(50)	Conditionally		
POSTAL_CODE	Varchar2(30)	Conditionally		
ADDRESS_ID	Number	Conditionally		
CUSTOMER_ITEM_NUMBER	Varchar2(50)	✓		
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	✓		
INACTIVE_FLAG	Varchar2(1)	✓		
ATTRIBUTE_CATEGORY	Varchar2(30)			
ATTRIBUTEn	Varchar2(150)			
ERROR_CODE	Varchar2(9)			
ERROR_EXPLANATION	Varchar2(2000)			

Table 3 – 20 List of Columns, Customer Item Cross–Reference Interface (Page 2 of 2)

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.

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 each of these Standard Who columns.

- 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.
- 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.
- 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.
- 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_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.

CHAPTER

4

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

This chapter contains information about the following Oracle Master Scheduling/MRP open 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.

MRP_FORECAST_INTERFACE Column Name	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	Number			
FORECAST_DESIGNATOR	Varchar2(10)	✓		
ORGANIZATION_ID	Number	✓		
FORECAST_DATE	Date	✓		
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number	✓		
LAST_UPDATE_LOGIN	Number			✓
QUANTITY	Number	✓		
PROCESS_STATUS	Number	✓		
CONFIDENCE_PERCENTAGE	Number	✓		
COMMENTS	Varchar2(240)			✓
ERROR_MESSAGE	Varchar2(240)		✓	
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
WORKDAY_CONTROL	Number			✓
BUCKET_TYPE	Number			✓
FORECAST_END_DATE	Date			✓(1)
TRANSACTION_ID	Number			✓
SOURCE_CODE	Varchar2(10)			✓

Table 4 – 1 Oracle Master Scheduling/MRP Open Forecast Interface (Page 1 of 2)

MRP_FORECAST_INTERFACE Column Name	Type	Required	Derived	Optional
SOURCE_LINE_ID	Number			✓
ATTRIBUTE_CATEGORY	Varchar2(30)			✓
ATTRIBUTE1 – ATTRIBUTE15	Varchar2(150)			✓

Table 4 – 1 Oracle Master Scheduling/MRP Open Forecast Interface (Page 2 of 2)

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 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 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 11.

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:

MRP_SCHEDULE_INTERFACE Column Name	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	Number	✓		
SCHEDULE_DESIGNATOR	Varchar2(10)	✓		
ORGANIZATION_ID	Number	✓		
LAST_UPDATE_DATE	Date			✓
LAST_UPDATED_BY	Number			✓
CREATION_DATE	Date			✓
CREATED_BY	Number			✓
LAST_UPDATE_LOGIN	Number			✓
SCHEDULE_DATE	Date	✓		
NEW_SCHEDULE_DATE	Date		✓	
RATE_END_DATE	Date			✓
NEW_RATE_END_DATE	Date		✓(1)	
SCHEDULE_QUANTITY	Number	✓		
SCHEDULE_COMMENTS	Varchar2(240)			✓
ERROR_MESSAGE	Varchar2(240)		✓	
WORKDAY_CONTROL	Number			✓
TRANSACTION_ID	Number			✓
PROCESS_STATUS	Number	✓		
SOURCE_CODE	Varchar2(10)			✓
SOURCE_LINE_ID	Number			✓
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	

Table 4 – 2 Oracle Master Scheduling/MRP Open Master Schedule Interface (Page 1 of 2)

MRP_SCHEDULE_INTERFACE Column Name	Type	Required	Derived	Optional
PROGRAM_UPDATE_DATE	Date		✓	
ATTRIBUTE_CATEGORY	Varchar2(30)			✓
ATTRIBUTE1 – ATTRIBUTE15	Varchar2(150)			✓

Table 4 – 2 Oracle Master Scheduling/MRP Open Master Schedule Interface (Page 2 of 2)

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 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 Master Scheduling/MRP also performs the following validation:

INVENTORY_ITEM_ID

Must be a valid item defined IN MTL_SYSTEM_ITEMS.

Master Demand Schedules can only include MPS planned or MRP planned items.

Master Production Schedules can only include MPS planned items.

ORGANIZATION_ID

Must be a valid organization defined in ORG_ORGANIZATION_DEFINITIONS.

SCHEDULE_DESIGNATOR

Must be a valid, non–disabled master schedule name defined in MRP_SCHEDULE_DESIGNATORS.

SCHEDULE_DATE

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

RATE_END_DATE

Must be greater than or equal to SCHEDULE_DATE.

Only repetitively planned items can have a RATE_END_DATE.

SCHEDULE_QUANTITY

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

WORKDAY_CONTROL

Must be one of:

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

PROCESS_STATUS

Must be one of:

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

TRANSACTION_ID

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

Resolving Failed Open Master Schedule Interface Rows

Error Messages

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

See Also

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

Viewing Failed Transactions

Use SQL*Plus 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 SQL*Plus 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 synchonous actions on the forecasting data, and you can manipulate data within a commit cycle controlled by the calling module.

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

Functional Overview

You can process MRP Forecast Entries directly from within your calling module without running a concurrent process, it is PL/SQL based. This program allows you to create new forecasts, replace existing forecasts, and delete forecast entries within a defined forecast name or designator. The forecast data that needs to be imported is loaded from a table and inserted into the MRP_FORECAST_DATES parameter.

Setting Up the Open Forecast Entries API

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

Inserting into the Open Forecast Entries API Tables

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

Open Forecast Entries Application Program Interface PL/SQL Table Description

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

T_FORECAST_INTERFACE Column Name	Type	Required	Derived	Optional
INVENTORY_ITEM_ID	Number	✓		
FORECAST_DESIGNATOR	Varchar2(10)	✓		
ORGANIZATION_ID	Number	✓		
FORECAST_DATE	Date	✓		
LAST_UPDATE_DATE	Number		✓	
CREATION_DATE	Date		✓	
CREATED_BY	Number		✓	
LAST_UPDATE_LOGIN	Number		✓	✓
QUANTITY	Number	✓		
PROCESS_STATUS	Number	✓		
CONFIDENCE_PRECENTAGE	Number	✓		
COMMENTS	Varchar2(240)			✓
ERROR_MESSAGE	Varchar2(240)		✓	
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
WORKDAY_CONTROL	Number			✓
BUCKET_TYPE	Number			✓

Table 4 – 3 Oracle Master Scheduling/MRP Open Forecast Entries API (Page 1 of 2)

T_FORECAST_INTERFACE Column Name	Type	Required	Derived	Optional
FORECAST_END_DATE	Date			✓
TRANSACTION_ID	Number			✓
SOURCE_CODE	Varchar2(10)			✓
SOURCE_LINE_ID	Number			✓
ATTRIBUTE1 – ATTRIBUTE15	Varchar2(150)			✓
PROJECT_ID	Number			✓
TASK_ID	Number			✓

Table 4 – 3 Oracle Master Scheduling/MRP Open Forecast Entries API (Page 2 of 2)

Open Forecast Interface Designator Table Description

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

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

Table 4 – 4 Oracle Master Scheduling/MRP Open Forecast Interface Designator (Page 1 of 1)

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 desginator 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

CHAPTER

5

Oracle Order Entry / Shipping Open Interfaces

This chapter contains information about the following Oracle Order Entry / Shipping open interfaces:

Integrating Oracle Order Entry/Shipping with Oracle Receivables

Oracle Order Entry/Shipping provides functionality to integrate with Oracle Receivables. Using AutoInvoice, you can create invoices, create credit memos and credits on account, recognize revenue, and manage sales credits.

Basic Needs

Oracle Order Entry/Shipping and Oracle Receivables provide features you need to satisfy the following integration needs:

- Create accurate and timely invoices, credit memos, and credits on account from order entry transactions
- Control when order transactions are invoiced

Major Features

Receivables Interface

Order Entry/Shipping provides a powerful program that automatically collects order and return information and populates the Oracle Receivables AutoInvoice interface tables. Using process parameters, you control which types of orders are interfaced each time you execute the program. Order Entry/Shipping ensures that all orders or order lines and returns or return lines have successfully completed any required prerequisites.

Invoice Source

The Invoice Source parameter for the Receivables Interface requires that an invoice source be set up with specific values for AutoInvoice validation. When defining invoice sources in Oracle Receivables, you must create at least one invoice source for Order Entry/Shipping's use if you want to interface orders and returns for processing by AutoInvoice.

The following table shows the necessary field values for the Transaction Sources window in Oracle Receivables.

Field in Transaction Sources Window	Necessary Value
Batch Source region	
Type	Imported
Status	Active
Automatic Transaction Numbering	Yes if *WSH: Invoice Numbering Method* is set to Automatic, No if set to Delivery Name
Customer Information region	
Sold–to Customer	Id
Bill–to Customer	Id
Bill–to Address	Id
Bill–to Contact	Id
Ship–to Customer	Id
Ship–to Address	Id
Ship–to Contact	Id
Payment Method Rule	(*any*)
Customer Bank Account	(*any*)
Accounting Information region	
Invoicing Rule	Id
Accounting Rule	Id
Accounting Flexfield	(*any*)
Derive Date	(*any*)
Payment Terms	Id
Revenue Account allocation	(*any*)
Other Information region	
Transaction Type	Id
Memo Reason	Id
Agreement	Id
Memo Line Rule	Id

Table 5 – 1 Transaction Sources Window: Required Settings (Page 1 of 2)

Field in Transaction Sources Window	Necessary Value
Other Information region	
Sales Territory	(*any*)
Inventory Item	Id
Unit of Measure	Id
FOB Point	Code
Freight Carrier	Code
Related Document	Id
Sales Credits Validation region	
Salesperson	Id
Sales Credit Type	Id
Sales Credit	Percent

Table 5 – 1 Transaction Sources Window: Required Settings (Page 2 of 2)

Note: In the Batch Source region, you must define at least one transaction source with automatic invoice numbering, regardless of your setting for the *WSH: Invoice Numbering Method* profile.

Automatic Tax Calculation

As orders from Order Entry / Shipping are processed, AutoInvoice automatically calculates sales tax based on the Sales Tax Location flexfield combination. If you have designated a customer as tax–exempt, AutoInvoice will not tax any items billed for the customer. Similarly, if you have designated an item as non–taxable, AutoInvoice will not tax the item.

Automatic Account Code Creation

Oracle Receivables uses AutoAccounting to determine the revenue account for all transactions from Order Entry / Shipping. AutoAccounting lets you define what information is used to define the various segments of your Accounting Flexfield.

Accounting and Invoicing Rules

Order Entry / Shipping uses accounting and invoicing rules. This information is transferred to Oracle Receivables and used to determine

the invoice date (invoicing rule) and general ledger distribution records (accounting rule). Order Entry / Shipping passes an invoicing rule and accounting rule for each order transaction interfaced to Oracle Receivables, except for when the accounting rule is **Immediate**, in which case Order Entry / Shipping does not pass any value (inserts null).

Accounting Rules

Order Entry / Shipping determines the accounting rule for sales order lines based on the following hierarchy.

Accounting Rule Hierarchy for Sales Order Lines	
1	If you referenced an agreement on the order that *does not* allow override of the accounting rule, Order Entry / Shipping inserts the accounting rule from the associated agreement (SO_AGREEMENTS. ACCOUNTING_RULE_ID); ***if not, then...***
2	If you referenced a commitment on the order line that is associated with an agreement that *does not* allow override of the accounting rule, Order Entry / Shipping inserts the accounting rule from the agreement (SO_AGREEMENTS.ACCOUNTING_ RULE_ID); ***if not, then...***
3	If you defined an accounting rule for the item, Order Entry / Shipping will use the accounting rule for the item (MTL_SYSTEM_ITEMS.ACCOUNTING_RULE_ID); ***if not, then...***
4	If you referenced a commitment on the order that is associated with an agreement that *does* allow override of the accounting rule, Order Entry / Shipping inserts the accounting rule from the associated agreement (SO_AGREEMENTS.ACCOUNTING_ RULE_ID); ***if not, then...***
5	If you referenced an agreement on the order line that *does* allow override of the accounting rule, Order Entry / Shipping inserts the accounting rule from the agreement (SO_AGREEMENTS. ACCOUNTING_RULE_ID); ***if not, then...***
6	In all other cases, Order Entry / Shipping inserts the accounting rule on the order.

Table 5 – 2 Accounting Rule Hierarchy for Sales Order Lines (Page 1 of 1)

Invoicing Rules

Order Entry / Shipping determines the invoicing rule for a sales order line based on the following hierarchy:

Invoicing Rule Hierarchy for Sales Order Lines	
1	If you referenced an agreement on the order that *does not* allow override of the invoicing rule, Order Entry / Shipping inserts the invoicing rule from the agreement (SO_AGREEMENTS. INVOICING_RULE_ID); ***if not, then...***
2	If you referenced a commitment on the order line that is associated with an agreement that *does not* allow override of the invoicing rule, Order Entry / Shipping inserts the invoicing rule from the agreement (SO_AGREEMENTS.INVOICING_RULE_ID); ***if not, then...***
3	If you defined an invoicing rule for the item, Order Entry / Shipping will use the invoicing rule for the item (MTL_SYSTEM_ITEMS.INVOICING_RULE_ID); ***if not, then...***
4	If you referenced a commitment on the order line that is associated with an agreement that *does* allow override of the invoicing rule, Order Entry / Shipping inserts the invoicing rule from that agreement (SO_AGREEMENTS.INVOICING_ RULE_ID); ***if not, then...***
5	If you referenced an agreement on the order that *does* allow override of the invoicing rule, Order Entry / Shipping inserts the invoicing rule from the agreement. (SO_AGREEMENTS. INVOICING_RULE_ID); ***if not, then...***
6	In all other cases, Order Entry / Shipping inserts the invoicing rule on the order.

Table 5 – 3 Invoicing Rule Hierarchy for Sales Order Lines (Page 1 of 1)

Credit Method for Accounting Rule

Order Entry / Shipping transfers a Credit Method for Accounting Rule for each return line. This credit method is recognized only by invoices that use *duration* accounting rules. You can assign a Credit Method for Accounting Rule to the order type of the return. If the Credit Method for Accounting Rule field for the order type is null, then Order Entry / Shipping transfers **LIFO** (Last In First Out).

Credit Method for Installments

Order Entry / Shipping transfers a Credit Method for Installments for each return line. This credit method is used for crediting an invoice that uses split payment terms. You can assign a Credit Method for Installments to the order type of the return. If the Credit Method for Installments field for the order type is null, then Order Entry / Shipping transfers **LIFO** (Last In First Out).

Internal Sales Orders

The Receivables Interface does not process internal sales order lines, even if it is a cycle action in the internal sales order's order cycle.

Internal sales orders are orders that originate in Oracle Purchasing as internal requisitions, and are imported to Order Entry / Shipping as internal sales orders using OrderImport.

See Also

Order Cycles, *Oracle Order Entry/Shipping User's Guide*

Using AutoAccounting, *Oracle Receivables User's Guide*

Accounting Rules, *Oracle Receivables User's Guide*

Entering Commitments, *Oracle Receivables User's Guide*

Defining Items, *Oracle Inventory User's Guide*

Defining Agreements, *Oracle Order Entry/Shipping User's Guide*

Accounting for Credit Memos, *Oracle Receivables User's Guide*

Invoicing of ATO Configurations

Invoicing Attributes

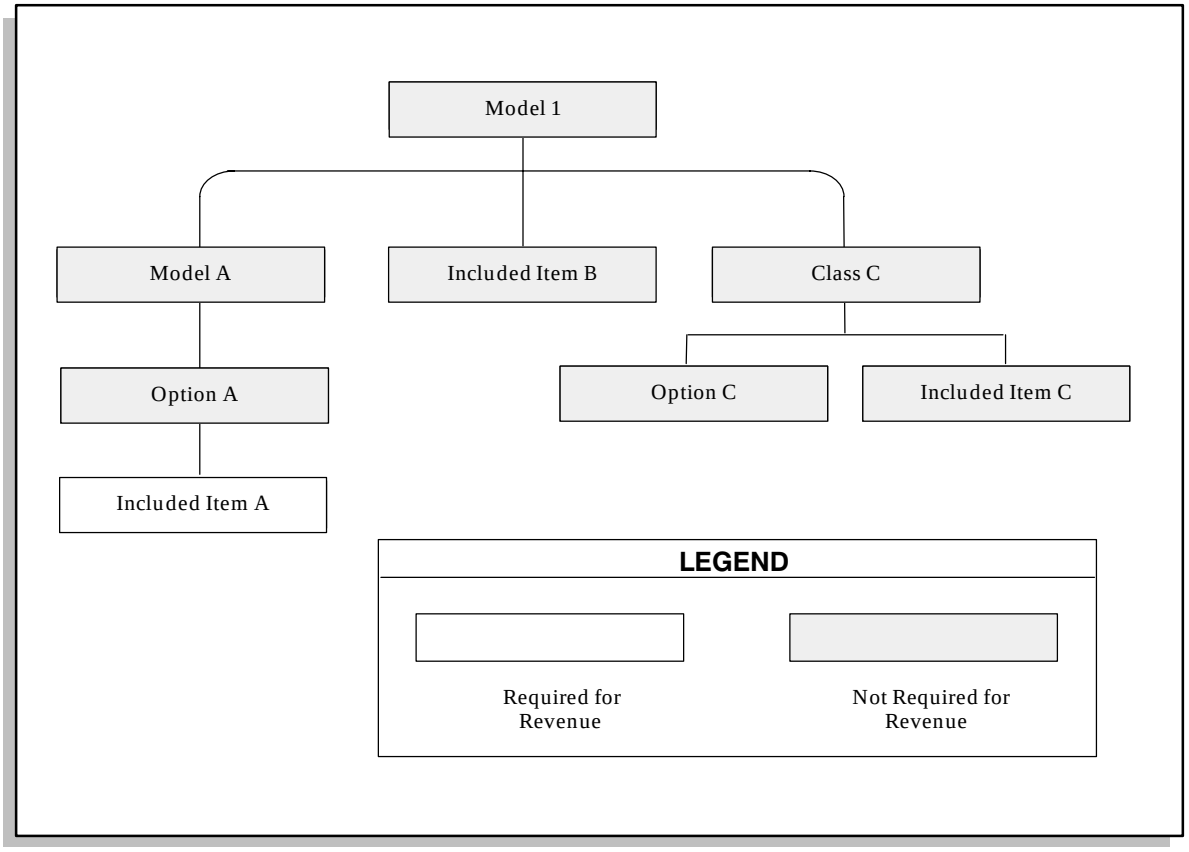
For ATO configurations, Order Entry / Shipping considers the base model's item attribute of a configuration to see if it should consider passing invoice information to Oracle Receivables for each order line in the configuration. If you have the item attributes **Invoiceable Item** and **Invoice Enabled** set to Yes for the base model item, Order Entry / Shipping then considers these item attributes for each component in the bill of material for the model to see if they should be invoiced in Oracle Receivables. If the item attributes **Invoiceable Item**

or **Invoice Enabled** are set to No for the base model item, Order Entry/Shipping does not pass invoicing information to Oracle Receivables for any order lines for the components within the configuration, regardless of the item attribute settings.

Required for Revenue Attribute

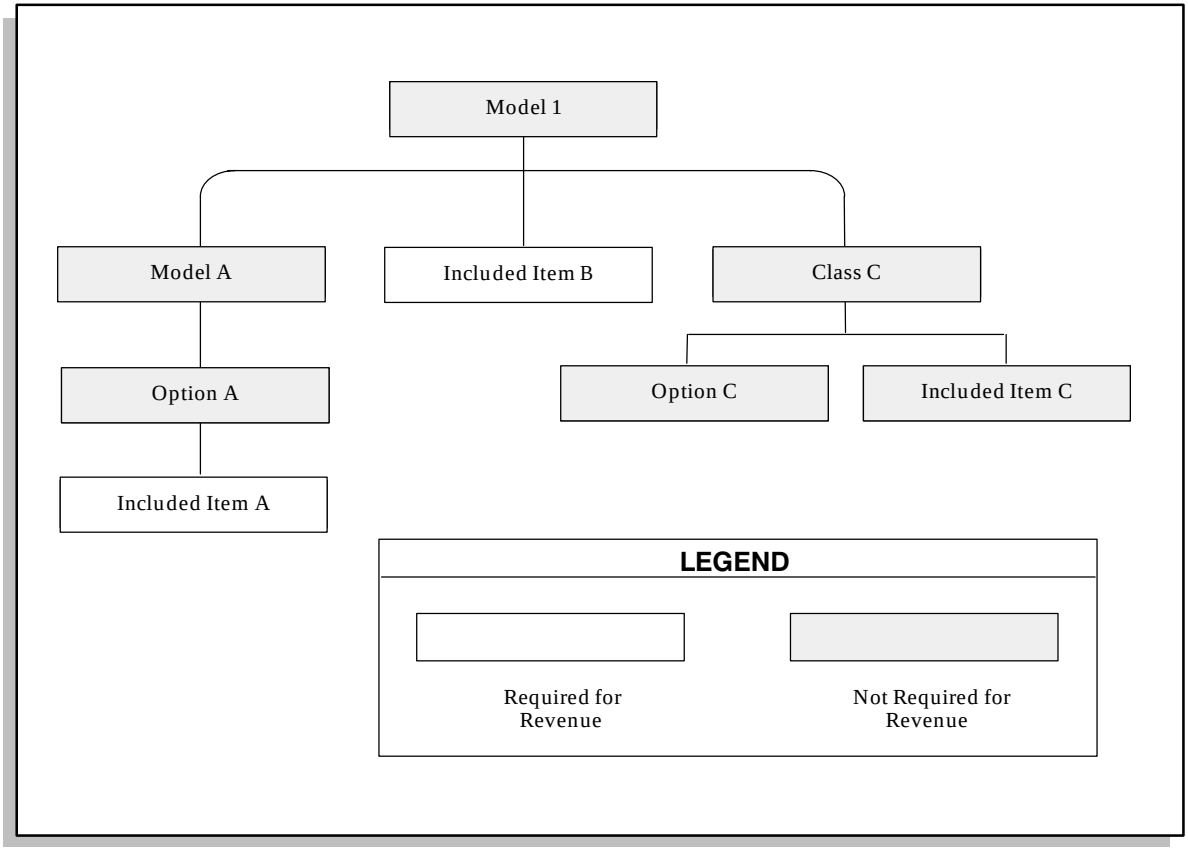
The bill of material attribute **Required for Revenue** allows you to define specific items in a bill that must be shipped before their parent can be invoiced. In all cases the control applies to only one level, the immediate parent. Except for classes, the control relationship is the child affecting the parent. The following diagrams demonstrate some examples.

Figure 5 – 1



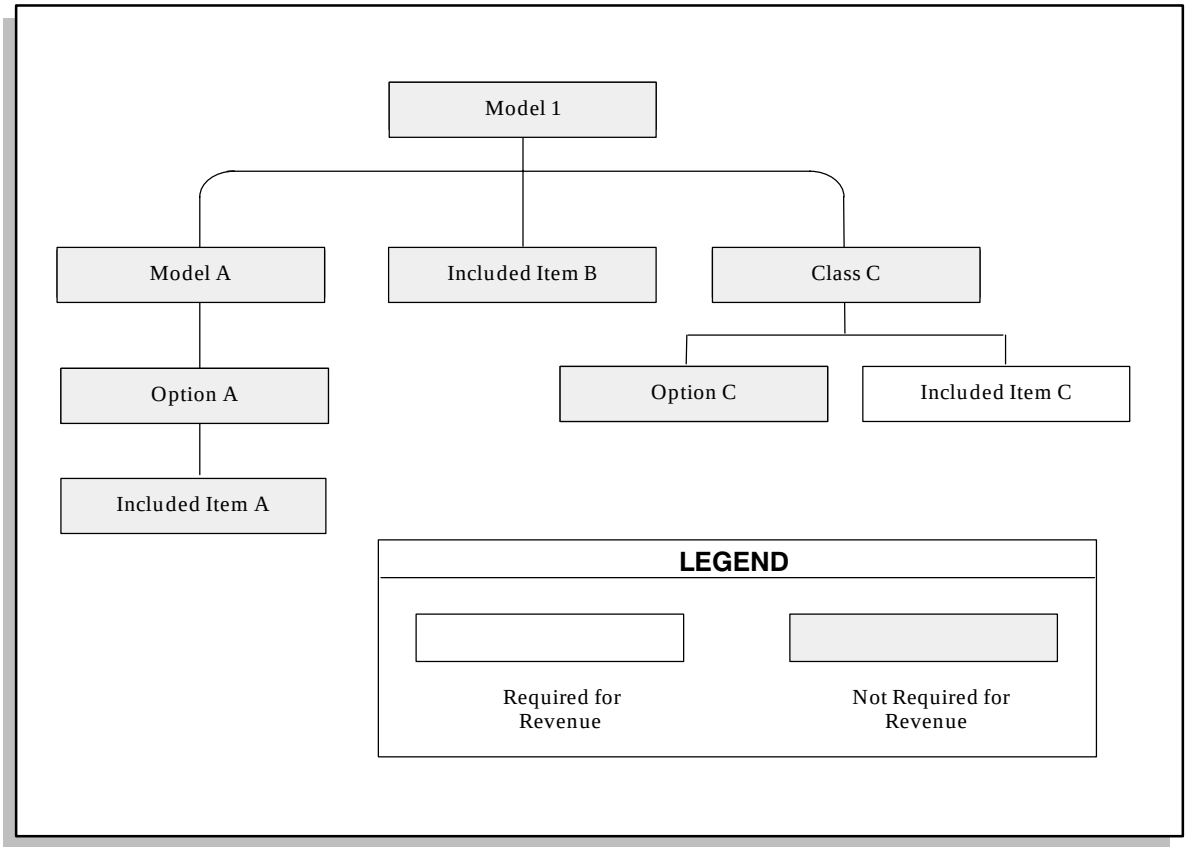
Here Included Item A has the Required for Revenue attribute set to Yes. Option A is not eligible to interface to Oracle Receivables until Included Item A is shipped, even if Option A is also shippable and has shipped. All other components, including Model A and Model 1, are eligible to interface regardless of Included Item A's shipment status.

Figure 5 – 2



Here Included Item B has the Required for Revenue attribute set to Yes. Model 1 is not eligible to interface to Oracle Receivables until Included Item B is shipped. And again, Option A is not eligible to interface until Included Item A is shipped.

Figure 5 – 3



The situation with classes is unique. If any item below a class in a bill has the Required for Revenue attribute set to Yes, then that item must be shipped before the parent item *and* the other items in the class are eligible to interface. For example, in the figure above, Included Item C has the Required for Revenue attribute set to Yes. Therefore, both Option C and Class C are not eligible to interface until Included Item C is shipped.

See Also

Item Attributes Used by Order Entry / Shipping, *Oracle Order Entry/Shipping User's Guide*

Overview of Bills of Material, *Oracle Bills of Material User's Guide*

Understanding the Interface Tables

Oracle Order Entry / Shipping inserts information into two of the three AutoInvoice interface tables (RA_INTERFACE_LINES and RA_INTERFACE_SALES_CREDITS). RA_INTERFACE_ DISTRIBUTIONS is not described in this essay because all account code creation is done by AutoInvoice based on the AutoAccounting rules you have defined. The following describes what information Order Entry / Shipping interfaces for each order and order line, each sales credit, and each freight charge.

RA_INTERFACE_LINES

INTERFACE_LINE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

INTERFACE_LINE_CONTEXT **VARCHAR2(30)**

Order Entry / Shipping inserts your value for the *OE: Source Code* profile option.

INTERFACE_LINE_ATTRIBUTE1 **VARCHAR2(30)**

Order Entry / Shipping inserts SO_HEADERS.ORDER_NUMBER.

INTERFACE_LINE_ATTRIBUTE2 **VARCHAR2(30)**

Order Entry / Shipping inserts SO_ORDER_TYPES.NAME.

INTERFACE_LINE_ATTRIBUTE3 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts WSH_DELIVERIES.NAME.

Non–shipped order line Order Entry / Shipping inserts 0 (zero).

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts WSH_DELIVERIES.NAME.

INTERFACE_LINE_ATTRIBUTE4 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts SUBSTR(WSH_DELIVERIES.WAYBILL, 1, 30).

Non–shipped order line Order Entry / Shipping inserts 0 (zero).

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts SUBSTR(WSH_DELIVERIES.WAYBILL, 1, 30).

INTERFACE_LINE_ATTRIBUTE5 **VARCHAR2(30)**

Sales order or return line Order Entry / Shipping inserts the number of times the order or return line has been interfaced for invoice or credit (SO_LINES.INVOICE_ COUNT+1).

Freight charges Order Entry / Shipping inserts 1.

INTERFACE_LINE_ATTRIBUTE6 **VARCHAR2(30)**

Sales order or return line Order Entry / Shipping inserts SO_LINES.LINE_ID.

Freight charges Order Entry / Shipping inserts 0 (zero).

INTERFACE_LINE_ATTRIBUTE7 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts SO_PICKING_LINES. PICKING_LINE_ID

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts 0 (zero).

INTERFACE_LINE_ATTRIBUTE8 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts WSH_DEPARTURES. BILL_OF_LADING.

Non–shipped order line Order Entry / Shipping inserts 0 (zero).

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts WSH_DEPARTURES. BILL_OF_LADING.

INTERFACE_LINE_ATTRIBUTE9 **VARCHAR2(30)**

Order Entry / Shipping inserts the customer item number, if one is defined. Otherwise, it inserts 0 (zero).

INTERFACE_LINE_ ATTRIBUTE10 **VARCHAR2(30)**

Order Entry / Shipping inserts WAREHOUSE_ID from SO_PICKING_LINES, SO_LINE_DETAILS, or SO_LINES.

INTERFACE_LINE_ ATTRIBUTE11–15 **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

BATCH_SOURCE_NAME Not Null **VARCHAR2(50)**

Order Entry / Shipping enters the invoice source name you select when you run the Receivables Interface program.

SET_OF_BOOKS_ID Not Null **NUMBER (15)**

Order Entry / Shipping inserts the ID from the *OE: Set of Books* profile setting.

LINE _TYPE Not Null **VARCHAR2(20)**

Sales order or return line Order Entry / Shipping inserts **LINE**.

Freight charges Order Entry / Shipping inserts **FREIGHT** for shipment freight charges.

DESCRIPTION Not Null **VARCHAR2(240)**

Sales order or return line Order Entry / Shipping inserts MTL_SYSTEM_ITEMS.DESCRIPTION for the item.

Freight charges Order Entry / Shipping inserts **Freight Charge** for shipment freight charges.

CURRENCY_CODE Not Null **VARCHAR2(30)**

Shipped order line or freight charges
Order Entry / Shipping inserts WSH_DELIVERIES.CURRENCY_CODE.

Non–shipped line or return line
Order Entry / Shipping inserts SO_HEADERS.CURRENCY_CODE.

AMOUNT **NUMBER**

Sales order or return line
Order Entry / Shipping inserts a calculated amount (SO_LINES. SELLING_PRICE multiplied by SO_LINES.QUANTITY) based on the calculated quantity. Order Entry / Shipping rounds the amount based on the minimum accounting unit and precision associated with the currency of the order.

The amount sign will match the sign on the quantity based on the value of RA_CUST_TRX_TYPES.CREATION_SIGN.

Freight charges
Order Entry / Shipping inserts

```
SUM(SO_FREIGHT_CHARGES.AMOUNT)
    FROM SO_FREIGHT_CHARGES
    GROUP BY SO_FREIGHT_CHARGES.
        PICKING_HEADER_ID).
```

CUST_TRX_TYPE_NAME **VARCHAR2(20)**

Order Entry / Shipping does not insert a value into this column.

CUST_TRX_TYPE_ID **NUMBER(15)**

Sales order line
Order Entry / Shipping inserts SO_ORDER_TYPES.CUST_ TRX_TYPE_ID for the order type associated with the order. However, if there is a commitment on the order line, Order Entry / Shipping does not insert a value into this column so that AutoInvoice can default the CUST_TRX_TYPE_ID from the commitment.

Return line
Order Entry / Shipping inserts

```
RA_CUST_TRX_TYPES.CREDIT_MEMO_TYPE_ID
        WHERE  RA_CUST_TRX_TYPES.CUST_TRX_TYPE_ID

                                =
SO_ORDER_TYPES.CUST_TRX_TYPE_ID
        AND  SO_ORDER_TYPES.ORDER_TYPE_ID
                        = SO_HEADERS.ORDER_TYPE_ID
```

AND SO_HEADERS.HEADER_ID
= SO_LINES.HEADER_ID.

Freight charges Order Entry / Shipping inserts SO_ORDER_TYPES.CUST_TRX_TYPE_ID associated with the order on the shipment.

TERM_NAME **VARCHAR2(15)**

Order Entry / Shipping does not insert a value into this column.

TERM_ID **NUMBER(15)**

Sales order line Order Entry / Shipping inserts SO_LINES.TERMS_ID.

Return line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts SO_HEADERS.TERMS_ID associated with the order on the shipment.

ORIG_SYSTEM_BILL_CUSTOMER_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_BILL_CUSTOMER_ID **NUMBER(15)**

Order Entry / Shipping inserts

RA_ADDRESSES.CUSTOMER_ID FROM RA_ADDRESSES
WHERE SO_HEADERS.INVOICE_TO_SITE_USE_ID
= RA_SITE_USES.SITE_USE_ID
AND RA_SITE_USES.ADDRESS_ID
= RA_ADDRESSES.ADDRESS_ID.

ORIG_SYSTEM_BILL_ADDRESS_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_BILL_ADDRESS_ID **NUMBER(15)**

Order Entry / Shipping inserts

RA_SITE_USES.ADDRESS_ID FROM RA_ADDRESSES
WHERE SO_HEADERS.INVOICE_TO_SITE_USE_ID
= RA_SITE_USES.SITE_USE_ID.

ORIG_SYSTEM_BILL_CONTACT_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_BILL_CONTACT_ID **NUMBER(15)**

Order Entry / Shipping inserts SO_HEADERS.INVOICE_TO_CONTACT_ID.

ORIG_SYSTEM_SHIP_CUSTOMER_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_SHIP_CUSTOMER_ID **NUMBER(15)**

Sales order or return line

Order Entry / Shipping inserts

RA_ADDRESSES.CUSTOMER_ID FROM RA_ADDRESSES WHERE RA_ADDRESSES.ADDRESS_ID = RA_SITE_USES.ADDRESS_ID AND RA_SITE_USES.SITE_USE_ID = NVL(SO_LINES.SHIP_TO_SITE_USE_ID, SO_HEADERS.SHIP_TO_SITE_USE_ID).

Freight charges

Order Entry / Shipping inserts

RA_ADDRESSES.CUSTOMER_ID FROM RA_ADDRESSES WHERE RA_ADDRESSES.ADDRESS_ID = RA_SITE_USES.ADDRESS_ID AND RA_SITE_USES.SITE_USE_ID = SO_HEADERS.SHIP_TO_SITE_USE_ID.

ORIG_SYSTEM_SHIP_ADDRESS_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_SHIP_ADDRESS_ID **NUMBER(15)**

Sales order or return line

Order Entry / Shipping inserts

RA_SITE_USES.ADDRESS_ID FROM RA_ADDRESSES WHERE RA_SITE_USES.SITE_USE_ID = NVL(SO_PICKING_LINES.SHIP_TO_SITE_USE_ID, NVL(SO_PICKING_HEADERS.SHIP_TO_SITE_USE_ID, NVL(SO_LINES.SHIP_TO_SITE_USE_ID, SO_HEADERS.SHIP_TO_SITE_USE_ID))).

Freight charges

Order Entry / Shipping inserts

RA_SITE_USES.ADDRESS_ID FROM RA_ADDRESSES WHERE RA_SITE_USES.SITE_USE_ID = NVL(SO_PICKING_HEADERS.SHIP_TO_SITE_USE_ID, SO_HEADERS.SHIP_TO_SITE_USE_ID).

ORIG_SYSTEM_SHIP_CONTACT_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_SHIP_CONTACT_ID **NUMBER(15)**

Sales order or return line

Order Entry / Shipping inserts
NVL(SO_LINES.SHIP_TO_CONTACT_ID, SO_HEADERS.SHIP_TO_CONTACT_ID).

Freight charges

Order Entry / Shipping inserts
SO_HEADERS.SHIP_TO_CONTACT_ID.

ORIG_SYSTEM_SOLD_CUSTOMER_REF **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

ORIG_SYSTEM_SOLD_CUSTOMER_ID **NUMBER(15)**

Order Entry / Shipping inserts SO_HEADERS.CUSTOMER_ID.

LINK_TO_LINE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

LINK_TO_LINE_CONTEXT **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

LINK_TO_LINE_ATTRIBUTE1 – 15 **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

RECEIPT_METHOD_NAME **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

RECEIPT_METHOD_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

CONVERSION_TYPE Not Null **VARCHAR2(30)**

Order Entry / Shipping inserts NVL(SO_HEADERS.CONVERSION_TYPE, 'User').

CONVERSION_DATE **DATE**

Order Entry / Shipping inserts SO_HEADERS.CONVERSION_DATE.

CONVERSION_RATE **NUMBER**

Order Entry / Shipping inserts SO_HEADERS.CONVERSION_RATE.

CUSTOMER_TRX_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

TRX_DATE **DATE**

Order Entry / Shipping does not insert a value into this column.

GL_DATE **DATE**

Order Entry / Shipping does not insert a value into this column.

DOCUMENT_NUMBER **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

TRX_NUMBER **VARCHAR2(20)**

If the *WSH: Invoice Numbering Method* profile is set to **Automatic**, AutoInvoice determines a unique number for this transaction. If the profile is set to **Delivery Name**, Order Entry / Shipping inserts a delivery name. An index is appended if the delivery has more than one invoice. For example, Order Entry / Shipping might insert 'delivery' for the first invoice, 'delivery–1' for the second, 'delivery–2' for the third, and so on.

LINE_NUMBER **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

QUANTITY **NUMBER**

Order Entry / Shipping inserts a calculated quantity based on the type of line being interfaced.

Sales order line The quantity will be either negative or positive, depending on the value of RA_CUST_TRX_TYPES.CREATION_SIGN associated with the invoice type for the sales order. If RA_CUST_TRX_TYPES. CREATION_SIGN is **N**, then quantity passed is –1 multiplied by the quantity calculated.

Shipped order line When picking lines exist for the order line:

STANDARD LINES

Quantity = SO_PICKING_LINES.SHIPPED_QUANTITY
– SO_PICKING_LINES.INVOICED_QUANTITY

PTO CONFIGURATION LINES – PTO MODEL

This is a multistep process:

1. Starting quantity = SO_LINES.SHIPPED_QUANTITY
   – SO_LINES.INVOICED_QUANTITY

2. Next, look at all the required for revenue included items for the model and the required for revenue ”children” lines of the model (option class, option items, models):

   min child quantity =
   min (child SO_LINES.QUANTITY_TO_INVOICE/
   child SO_LINES.ORDERED_QUANTITY)
   * model SO_LINES.ORDERED_QUANTITY
   where the child line items are required
   for revenue in the bill of material for the model

   min included item quantity =
   min (included item
SO_PICKING_LINES.SHIPPED_QUANTITY/
   included item
SO_PICKING_LINES.ORIGINAL_REQUESTED_
   QUANTITY)* model SO_LINES.ORDERED_QUANTITY
   where the included items are required for revenue in the bill of material for the model

3. Total quantity to invoice =
   minimum of (STARTING_QUANTITY,
   min child quantity,
   min included item quantity)

4. Quantity =
   minimum of (SO_PICKING_LINES.SHIPPED_QUANTITY
   for the model, total quantity to invoice)

– PTO CLASS

1. Starting quantity = SO_LINES.SHIPPED_QUANTITY
   – SO_LINES.INVOICED_QUANTITY

2. Next, look at all the required for revenue included items for the class and the required for revenue ”children” lines of the class (option class, option items)

   min child quantity =
   min (child SO_LINES.QUANTITY_TO_INVOICE/
   child SO_LINES.ORDERED_QUANTITY)
   * class SO_LINES.ORDERED_QUANTITY
   where the child line items are required
   for revenue in the bill of material for the class

   min included item quantity =
   min (included item

SO_PICKING_LINES.SHIPPED_QUANTITY /
included item
SO_PICKING_LINES.ORIGINAL_REQUESTED_
QUANTITY) * model SO_LINES.ORDERED_QUANTITY
where the included items are required for revenue in the BOM for the class

3. Total quantity to invoice = minimum of (STARTING_QUANTITY, min child quantity, min included item quantity)

4. Quantity =
minimum of (SO_PICKING_LINES.SHIPPED_QUANTITY for the class, total quantity to invoice)

– PTO OPTION ITEM OR KIT

1. Starting quantity = SO_LINES.SHIPPED_QUANTITY – SO_LINES.INVOICED_QUANTITY

2. Next, look at all the required for revenue included items for the option item and the required for revenue included items of the option item ”parent” line (option class)

min included item quantity =
min (included item
SO_PICKING_LINES.SHIPPED_QUANTITY /
included item
SO_PICKING_LINES.ORIGINAL_REQUESTED_
QUANTITY) * option item SO_LINES.ORDERED_QUANTITY
where the included items are required for revenue in the bill of material for the option item or for the option item ”parent”class

3. Total quantity to invoice = minimum of (STARTING_QUANTITY, min included item quantity)

4. Quantity = minimum of
(SO_PICKING_LINES.SHIPPED_QUANTITY
for the class, total quantity to invoice)

ATO CONFIGURATION LINES

– ATO MODEL

Quantity =
SO_PICKING_LINES.SHIPPED_QUANTITY
– SO_PICKING_LINES.INVOICED_QUANTITY

– ATO CLASS

Quantity =
(class ORDERED_QUANTITY / model ordered quantity)
* model quantity

– ATO OPTION ITEM

Quantity =
(option item ordered quantity / model ordered quantity)
* model quantity

Non–shipped order line

Non–Ship Cycle or Receivables Interface prior to Picking / Shipping:

Quantity = SO_LINES.ORDERED_QUANTITY
– SO_LINES.CANCELLED_QUANTITY

Return line

Order Entry / Shipping uses ORDERED_QUANTITY, SHIPPED_QUANTITY and INVOICED_QUANTITY to hold the values for the quantity authorized to be returned, the quantity received and accepted in Oracle Inventory and the quantity credited in Oracle Receivables, respectively. The value of SO_LINES.SHIPPED_QUANTITY is calculated in the RMA Inventory Interface program.

The quantity will be either negative or positive depending on the value of RA_CUST_TRX_TYPES.CREATION_SIGN associated with the credit memo type tied to the invoice type for return. If RA_CUST_TRX_TYPES.CREATION_SIGN is **A** or **N**, then the quantity passed is –1 multiplied by the quantity calculated.

Return line with receipt verification

With RMA Inventory Receipt

Quantity = SO_LINES.SHIPPED_QUANTITY –
SO_LINES.INVOICED_QUANTITY

Return line without receipt verification

Without RMA Inventory Receipt

Quantity = SO_LINES.ORDERED_QUANTITY –
SO_LINES.CANCELLED_QUANTITY

QUANTITY_ORDERED **NUMBER**

Sales order line or return line

Order Entry / Shipping inserts SO_LINES.ORDERED_QUANTITY.

Freight charges

Order Entry / Shipping does not insert a value into this column.

UNIT_SELLING_PRICE **NUMBER**

Sales Order lines and Return lines Order Entry / Shipping inserts SO_LINES.SELLING_PRICE.

Freight charges Order Entry / Shipping does not insert a value into this column.

UNIT_STANDARD_PRICE **NUMBER**

Sales order line or return line Order Entry / Shipping inserts SO_LINES.LIST_PRICE.

Freight charges Order Entry / Shipping does not insert a value into this column.

PRINTING_OPTION **VARCHAR2(20)**

Order Entry / Shipping does not insert a value into this column.

INTERFACE_STATUS **VARCHAR2(1)**

Order Entry / Shipping does not insert a value into this column.

REQUEST_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

RELATED_BATCH_ SOURCE_NAME **VARCHAR2(50)**

Order Entry / Shipping does not insert a value into this column.

RELATED_TRX_NUMBER **VARCHAR2(20)**

Order Entry / Shipping does not insert a value into this column.

RELATED_CUSTOMER_ TRX_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

PREVIOUS_CUSTOMER_ TRX_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

CREDIT_METHOD_FOR_ACCT_RULE **VARCHAR2(30)**

Sales order line Order Entry / Shipping does not insert a value into this column.

Return line Order Entry / Shipping inserts the accounting credit method, NVL(SO_ORDER_TYPES.ACCOUNTING_CREDIT_METHOD_CODE, 'LIFO') associated with the order type.

CREDIT_METHOD_FOR_INSTALLMENTS **VARCHAR2(30)**

Sales order line Order Entry / Shipping does not insert a value into this column.

Return line Order Entry / Shipping inserts the invoicing credit method, NVL(SO_ORDER_TYPES.INVOICING_CREDIT_METHOD_CODE, 'LIFO') associated with the order type.

REASON_CODE **VARCHAR2(30)**

Sales order line Order Entry / Shipping does not insert a value into this column.

Return line Order Entry / Shipping inserts SO_LINES.TRANSACTION_REASON_CODE.

TAX_RATE **NUMBER**

Order Entry / Shipping does not insert a value into this column.

TAX_CODE **VARCHAR2(50)**

Order Entry / Shipping does not insert a value into this column.

TAX_PRECEDENCE **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

EXCEPTION_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

EXEMPTION_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

SHIP_DATE_ACTUAL **DATE**

Shipped order line Order Entry / Shipping inserts SO_PICKING_HEADERS. DATE_SHIPPED.

Non–shipped order line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts SO_PICKING_HEADERS. DATE_SHIPPED.

FOB_POINT **VARCHAR2(20)**

Order Entry / Shipping only populates this column if the order line being invoiced has been shipped.

Shipped order line Order Entry / Shipping inserts SO_HEADERS.FOB_CODE.

Non–shipped order line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts

```
SO_HEADERS.FOB_CODE
FROM SO_HEADERS, SO_PICKING_HEADERS,
     SO_FREIGHT_CHARGES
WHERE SO_HEADERS.HEADER_ID
    = SO_PICKING_HEADERS.ORDER_HEADER_ID
AND SO_PICKING_HEADERS.PICKING_HEADER_ID
    = SO_FREIGHT_CHARGES.PICKING_HEADER_ID.
```

SHIP_VIA **VARCHAR2(20)**

Shipped order line Order Entry / Shipping inserts SO_PICKING_HEADERS.SHIP_METHOD_CODE.

Non–shipped order line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts SO_PICKING_HEADERS.SHIP_METHOD_CODE.

WAYBILL_NUMBER **VARCHAR2(50)**

Shipped order line Order Entry / Shipping inserts WSH_DELIVERIES.WAYBILL.

Non–shipped order line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts WSH_DELIVERIES.WAYBILL.

INVOICING_RULE_NAME **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

INVOICING_RULE_ID **NUMBER(15)**

Sales order line Order Entry / Shipping inserts DECODE(ACCOUNTING_RULE_ID,1, NULL,INVOICING_RULE_ID).

> **Note:** For more information on how Manufacturing, Distribution, Sales and Service determines the accounting rule used in the decode, see: Accounting and Invoicing Rules: page 5 – 4.

Return line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts DECODE(ACCOUNTING_RULE_ID,1, NULL,SO_HEADERS.INVOICING_RULE_ID).

ACCOUNTING_RULE_NAME **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

ACCOUNTING_RULE_ID **NUMBER(15)**

Sales order line Order Entry / Shipping inserts DECODE(ACCOUNTING_RULE_ID,1, NULL,ACCOUNTING_RULE_ID).

> **Note:** For more information on how Manufacturing, Distribution, Sales and Service determines the accounting rule used in the decode, see: Accounting and Invoicing Rules: page 5 – 4.

Return line Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping inserts DECODE(ACCOUNTING_RULE_ID,1, NULL,SO_HEADERS.ACCOUNTING_RULE_ID).

ACCOUNTING_RULE_ DURATION **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

RULE_START_DATE **DATE**

Order Entry / Shipping does not insert a value into this column.

PRIMARY_SALESREP_ NUMBER **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

PRIMARY_SALESREP_ID **NUMBER(15)**

Order Entry / Shipping inserts SO_HEADERS.SALESREP_ID.

SALES_ORDER **VARCHAR2(50)**

Order Entry / Shipping inserts SO_HEADERS.ORDER_NUMBER.

SALES_ORDER_LINE **VARCHAR2(30)**

Sales order or return line

Order Entry / Shipping inserts SO_LINES.LINE_NUMBER.

Freight charges

Order Entry / Shipping does not insert a value into this column.

SALES_ORDER_DATE **DATE**

Order Entry / Shipping inserts SO_HEADERS.DATE_ORDERED.

SALES_ORDER_SOURCE **VARCHAR2(50)**

Order Entry / Shipping inserts your value for the *OE: Source Code* profile option.

SALES_ORDER_REVISION **NUMBER**

Order Entry / Shipping does not insert a value into this column.

PURCHASE_ORDER **VARCHAR2(50)**

Order Entry / Shipping inserts SO_HEADERS.PURCHASE_ORDER_ NUM.

PURCHASE_ORDER_REVISION **VARCHAR2(50)**

Order Entry / Shipping does not insert a value into this column.

PURCHASE_ORDER_DATE **DATE**

Order Entry / Shipping does not insert a value into this column.

AGREEMENT_NAME **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

AGREEMENT_ID **NUMBER(15)**

Sales order line Order Entry / Shipping inserts SO_HEADERS.AGREEMENT_ID.

Return line For return lines with a purchase order or sales order reference, Order Entry / Shipping inserts the AGREEMENT_ID from the referenced order header. For return lines with an invoice reference, Order Entry / Shipping does not insert a value into this column.

Freight charges Order Entry / Shipping does not insert a value into this column.

MEMO_LINE_NAME **VARCHAR2(50)**

Order Entry / Shipping does not insert a value into this column.

MEMO_LINE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

INVENTORY_ITEM_ID **NUMBER(15)**

Sales order or return line Order Entry / Shipping inserts SO_LINES.INVENTORY_ITEM_ID.

Freight charges Order Entry / Shipping does not insert a value into this column.

MTL_SYSTEM_ITEMS_SEG1–20 **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

REFERENCE_LINE_ID **NUMBER(15)**

Sales order line Order Entry / Shipping inserts SO_LINES.COMMITMENT_ID.

Return line Order Entry / Shipping inserts SO_LINES.CREDIT_TO_LINE_ID.

Freight charges Order Entry / Shipping does not insert a value into this column.

REFERENCE_LINE_CONTEXT **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

REFERENCE_LINE_ ATTRIBUTE1–15 **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

TERRITORY_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

TERRITORY_SEGMENT1–20 **VARCHAR2(25)**

Order Entry / Shipping does not insert a value into this column.

ATTRIBUTE_CATEGORY **VARCHAR2(30)**

Sales order or return line Order Entry / Shipping inserts SO_LINES.CONTEXT.

Freight charges Order Entry / Shipping inserts SO_FREIGHT_CHARGES.CONTEXT.

ATTRIBUTE1–15 **VARCHAR2(150)**

Sales order or return line Order Entry / Shipping inserts SO_LINES.ATTRIBUTE1–15.

Freight charges Order Entry / Shipping inserts SO_FREIGHT_CHARGES. ATTRIBUTE1–15.

HEADER_ATTRIBUTE_CATEGORY **VARCHAR2(30)**

Sales order or return line Order Entry / Shipping inserts SO_HEADERS.CONTEXT.

Freight charges Order Entry / Shipping does not insert a value into this column.

HEADER_ATTRIBUTE1–15 **VARCHAR2(150)**

Sales order or return line Order Entry / Shipping inserts SO_HEADERS.ATTRIBUTE1–15.

Freight charges Order Entry / Shipping does not insert a value into this column.

COMMENTS **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

INTERNAL_NOTES **VARCHAR2(240)**

Order Entry / Shipping does not insert a value into this column.

INITIAL_CUSTOMER_TRX_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

USSGL_TRANSACTION_CODE_CONTEXT **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

USSGL_TRANSACTION_CODE **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

ACCTD_AMOUNT **NUMBER**

Order Entry / Shipping does not insert a value into this column.

CUSTOMER_BANK_ACCOUNT_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

CUSTOMER_BANK_ACCOUNT_NAME **VARCHAR2(25)**

Order Entry / Shipping does not insert a value into this column.

UOM_CODE **VARCHAR2(3)**

Sales order or return line Order Entry / Shipping inserts SO_LINES.UNIT_CODE.

Freight charges Order Entry / Shipping does not insert a value into this column.

UOM_NAME **VARCHAR2(25)**

Order Entry / Shipping does not insert a value into this column.

DOCUMENT_NUMBER_ SEQUENCE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

REASON_CODE_NAME **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

VAT_TAX_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

LOCATION_RATE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

REASON_CODE_MEANING **VARCHAR2(80)**

Order Entry / Shipping does not insert a value into this column.

LAST_PERIOD_TO_CREDIT **NUMBER**

Order Entry / Shipping does not insert a value into this column.

PAYING_CUSTOMER_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

PAYING_SITE_USE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

TAX_EXEMPT_FLAG **VARCHAR2(1)**

Order Entry / Shipping inserts SO_HEADERS.TAX_EXEMPT_FLAG for order lines.

SALES_TAX_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

CREATED_BY **NUMBER(15)**

Manufacturing, Distribution, Sales and Service enters an identification number to identify the user who created the record.

Validation: None

CREATION_DATE **DATE**

Manufacturing, Distribution, Sales and Service enters the creation date.

Validation: Standard Date Validation

LAST_UPDATED_BY **NUMBER(15)**

Manufacturing, Distribution, Sales and Service enters an identification number to identify the user who created or who most recently modified the record.

Validation: None

LAST_UPDATE_DATE **DATE**

Manufacturing, Distribution, Sales and Service enters the current date when a record is updated.

Validation: Standard Date Validation

LOCATION_SEGMENT_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

TAX_EXEMPT_REASON_CODE **VARCHAR2(30)**

Order Entry / Shipping inserts SO_HEADERS.TAX_EXEMPT_REASON_ CODE for order lines.

TAX_EXEMPT_NUMBER **VARCHAR2(80)**

Order Entry / Shipping inserts SO_HEADERS.TAX_EXEMPT_NUM for order lines.

TAX_EXEMPT_ REASON_CODE_MEANING **VARCHAR2(80)**

Order Entry / Shipping does not insert a value into this column.

RA_INTERFACE_SALESCREDITS

Manufacturing, Distribution, Sales and Service inserts one row for each sales credit row according to the following hierarchy:

1. Insert sales credits associated with the line; if none exists but the line is part of a configuration (ITEM_TYPE_CODE is **CLASS**, **KIT** or **STANDARD** and OPTION_FLAG is **Y**), then...
2. Insert sales credits associated with the model "parent" line; if none exists, then...
3. Insert sales credits associated with the order header.

INTERFACE_ SALESCREDIT_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

INTERFACE_LINE_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

INTERFACE_LINE_CONTEXT **VARCHAR2(30)**

Order Entry / Shipping inserts your value for the *OE: Source Code* profile option.

INTERFACE_LINE_ATTRIBUTE1 **VARCHAR2(30)**

Order Entry / Shipping inserts SO_HEADERS. ORDER_NUMBER.

INTERFACE_LINE_ATTRIBUTE2 **VARCHAR2(30)**

Order Entry / Shipping inserts SO_ORDER_TYPES.NAME.

INTERFACE_LINE_ATTRIBUTE3 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts WSH_DELIVERIES.NAME.

Non–shipped order line Order Entry / Shipping inserts 0 (zero).

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts WSH_DELIVERIES.NAME.

INTERFACE_LINE_ATTRIBUTE4 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts SUBSTR(WSH_DELIVERIES.WAYBILL, 1, 30).

Non–shipped order line Order Entry / Shipping inserts 0 (zero).

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts SUBSTR(WSH_DELIVERIES.WAYBILL, 1, 30).

INTERFACE_LINE_ATTRIBUTE5 **VARCHAR2(30)**

Order Entry / Shipping inserts the number of times the order, freight charge, or return line has been interfaced for invoice or credit.

INTERFACE_LINE_ATTRIBUTE6 **VARCHAR2(30)**

Sales order or return line Order Entry / Shipping inserts SO_LINES.LINE_ID.

Freight charges Order Entry / Shipping inserts 0 (zero).

INTERFACE_LINE_ATTRIBUTE7 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts SO_PICKING_LINES.PICKING_LINE_ID.

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts 0 (zero).

INTERFACE_LINE_ATTRIBUTE8 **VARCHAR2(30)**

Shipped order line Order Entry / Shipping inserts WSH_DEPARTURES.BILL_OF_LADING.

Non–shipped order line Order Entry / Shipping inserts 0 (zero).

Return line Order Entry / Shipping inserts 0 (zero).

Freight charges Order Entry / Shipping inserts WSH_DEPARTURES.BILL_OF_LADING.

INTERFACE_LINE_ATTRIBUTE9 **VARCHAR2(30)**

Order Entry / Shipping inserts the customer item number, if one is defined. Otherwise, it inserts 0 (zero).

INTERFACE_LINE_ATTRIBUTE10 **VARCHAR2(30)**

Order Entry / Shipping inserts WAREHOUSE_ID from SO_PICKING_LINES, SO_LINE_DETAILS, or SO_LINES.

INTERFACE_LINE_ATTRIBUTE11–15 **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

SALESREP_NUMBER **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

SALESREP_ID **NUMBER(15)**

Order Entry / Shipping inserts SO_SALES_CREDIT.SALESREP_ID.

SALES_CREDIT_TYPE_NAME **VARCHAR2(30)**

Order Entry / Shipping does not insert a value into this column.

SALES_CREDIT_TYPE_ID **NUMBER(15)**

Order Entry / Shipping inserts SO_SALES_CREDIT.SALES_CREDIT_TYPE_ID.

SALES_CREDIT_AMOUNT_SPLIT **NUMBER**

Order Entry / Shipping does not insert a value into this column.

SALES_CREDIT_PERCENT_SPLIT **NUMBER**

Order Entry / Shipping inserts SO_SALES_CREDIT.PERCENT.

INTERFACE_STATUS **VARCHAR2(1)**

Order Entry / Shipping does not insert a value into this column.

REQUEST_ID **NUMBER(15)**

Order Entry / Shipping does not insert a value into this column.

ATTRIBUTE_CATEGORY **VARCHAR2(30)**

Order Entry / Shipping inserts SO_SALES_CREDIT.CONTEXT.

ATTRIBUTE1–15 **VARCHAR2(150)**

Order Entry / Shipping inserts SO_SALES_CREDIT.ATTRIBUTE1–15.

Integrating Oracle Order Entry/Shipping Using OrderImport

With OrderImport, you can centralize your information by importing order entry data from a variety of sources, including both Oracle and non–Oracle systems. Order Entry / Shipping checks all of your data during the import process to ensure its validity within Order Entry / Shipping, then converts your import data into orders with lines, schedule details, price adjustments, and sales credits. Each time you run OrderImport, Order Entry / Shipping produces a report informing you of the total number of orders, order lines, price adjustments, and so on that OrderImport evaluated, and which succeeded or failed. You can use the OrderImport Process Exception Report to examine your data if it fails the import process.

Basic Needs

OrderImport provides the features you need to satisfy the following basic integration needs:

- Import data from a variety of environments, including your own order entry systems.
- Import complete (historical) orders.
- Import incomplete orders and finish them using the Sales Orders window.
- Import booked orders.
- Import changes to existing orders.
- Place demand or reserve inventory for orders automatically.
- Optionally price order lines using the same pricing rules as online entry, supporting individual customer price lists and discounts.
- Import quota and nonquota sales credit information for orders and order lines.
- Enforce holds and perform credit checking on any imported orders or changes.
- Import line schedule detail information with each order and order line.
- Enforce security rules and utilize defaulting with imported orders.

- Import internal requisition orders from Oracle Purchasing.
- Apply automatic notes.
- Review your imported data in a report. Not only can you identify which data has been successfully imported, but you can identify any errors which may have occurred during the import process.

Major Features

Order Importing

OrderImport lets you capture order data using a specialized system and import them into Oracle Order Entry/Shipping for processing. You can import orders with any entry status, including **Booked**. Imported orders can be queried and modified using the Sales Orders window in Order Entry/Shipping. If an order is imported with an entry status of Booked, it will automatically be eligible to progress to the next step of its order cycle when the import is complete.

Historical Data Conversion (Complete Orders)

OrderImport also allows you to import completed orders from your previous order entry system, making the transition from your old application to Order Entry/Shipping as smooth as possible. Complete orders are considered for historical purposes only and can have any entered state; the status for the **Complete Order** cycle action is **Closed**. Specify complete orders by entering **Y** in SO_HEADERS_INTERFACE.COMPLETE_FLAG.

Entered State

You can import both current and complete orders with any valid entry status. OrderImport ensures that all required fields for entry or booking are validated appropriately as the orders are imported. If any of the required fields for a booked order being imported is not supplied, Oracle Order Entry/Shipping rejects the order. See: Reporting: page 5 – 43.

Order Cycles

You can import an order within any valid order cycle. The order must be at the initial action of Enter or the final action of Complete. For

example, you cannot import an order that has been pick released but not yet closed.

Configurations

You can import all orderable items of a configuration, such as models, classes, and options. OrderImport, using the information you supply in the interface tables, will validate the configuration against the bill of material and create related order lines for the model, classes, and options.

If you are importing assemble–to–order (ATO) configurations, you cannot import the mandatory standard components required to build the configuration. Oracle Work in Process will select these automatically, from the bill of material, when the work order is opened. If you are importing pick–to–order (PTO) configurations, you cannot import the included items. Oracle Order Entry / Shipping will select these automatically, from the bill of material, when the included items are exploded. The point in the cycle when included items are exploded is defined by the profile option *OE: Included Item Freeze Method*.

You can also use match and reserve functionality with OrderImport when importing a booked order that contains an ATO model. Your ability to use this is controlled by the profile option *BOM: Check for Duplicate Configurations* and by the value in the SCHEDULE_STATUS_CODE column in the SO_HEADERS_INTERFACE table.

Changes to Imported Orders

You can import changes to orders that have already been imported by passing all changed and unchanged information through OrderImport. You can insert, update, and delete orders, order lines, price adjustments, and sales credits.

Order Scheduling

OrderImport lets you demand or reserve orders at the order level as they are imported, using the same rules as online order entry. If the scheduling request is unsuccessful on an imported order with an entry status of **Booked**, the order will be imported, and the scheduling exception is reported in the OrderImport Process Exception Report. You can then update the order in the Sales Orders or Schedule Orders windows. See: Overview of Order Scheduling, *Oracle Order Entry/Shipping User's Guide.*

Schedule Details

OrderImport allows you to import scheduling details for an order line by inserting data into the SO_LINE_DETAILS_INTERFACE table. For example, you may want to specify a subinventory or lot when you reserve an order line, or you can specify multiple detail records to split an order line between different warehouses or schedule dates.

Pricing

You can indicate whether you want to enter prices manually for imported orders or have Order Entry/Shipping automatically price order lines using the CALCULATE_PRICE column in the SO_LINES_INTERFACE table. We recommend that you use either completely automatic pricing or completely manual pricing with your imported orders. If you want to use the automatic method, you should set SO_LINES_INTERFACE.CALCULATE_PRICE to Yes, and define all your discounts as line–level and automatic. If you want to use the manual method, set SO_LINES_INTERFACE.CALCULATE_PRICE to No and import order lines with the list price, selling price, and any price adjustments. In this case, you should define all your discounts as line–level, overridable, and not automatic.

If you are using the manual method, Order Entry/Shipping lets you adjust the selling price of items through OrderImport by inserting data in the SO_PRICE_ADJUSTMENTS_INTERFACE table. If you are importing an order line whose list and selling prices are not the same, you *must* insert a valid price adjustment. The discount amount or percent of the referenced discount must either equal the difference between list and selling, or have Allow Override set to Yes. You can enter multiple price adjustments for an order line as long as each price adjustment uses a unique discount and the total of the price adjustments equals the difference between the list and selling price.

Attention: Line–level discounts that are defined as automatic are automatically applicable to imported orders that have Yes in the CALCULATE_PRICE column, so if you use automatic discounts, you do not need to import them with such order lines.

CALCULATE_PRICE – YES		CALCULATE_PRICE – NO	
List Price	Ignored	List Price	Required
Selling Price	Ignored	Selling Price	Required
Discount	Optional	Discount	Required if List Price ≠ Selling Price
Note: *Imported discounts must be overridable if you want to import a percent value that is different from that defined on the discount.*			

Table 5 – 4 Automatic and Manual Pricing Settings (Page 1 of 1)

Taxation and Tax Exemptions

OrderImport allows you to indicate the tax status of orders as they are imported, using the same rules as online order entry through the TAX_EXEMPT_FLAG, TAX_EXEMPT_NUM, and TAX_EXEMPT_REASON_CODE columns in the SO_HEADERS_INTERFACE table. The TAX_EXEMPT_FLAG allows you to indicate one of the following:

- This order is exempt for a normally taxable customer site and / or item
- Taxable for a normally non–taxable customer and / or item
- Taxation should be based on existing exemption rules

If you choose to exempt the order from taxation, you can enter a valid certificate number for the ship–to customer or enter a new, unapproved exemption certificate number in the TAX_EXEMPT_NUM column. If you choose to exempt the order from taxation, you must enter a TAX_EXEMPT_REASON_CODE. You can override the default tax code for an order line by specifying a value for TAX_CODE in the SO_LINES_INTERFACE table. If OrderImport determines that the order line is not taxable, the TAX_CODE column value in SO_LINES_INTERFACE will be preserved, but the imported order line will not have a tax code.

Sales Credits

Order Entry / Shipping lets you enter sales credit information through OrderImport by inserting data in the SO_SALES_CREDITS_INTERFACE table. If the salesperson in SO_HEADERS_INTERFACE.SALESREP_ID receives 100% sales credit

for an open order, you do not need to insert sales credits. However, if you want to apply sales credits to historical (closed) orders, split sales credits, or assign different credits to each order line, you can use the SO_SALES_CREDITS_INTERFACE table to communicate this information.

The total of the revenue credit must equal 100 percent. Nonrevenue credit can be any percentage of the order line or order.

Agreements

You can include an agreement name if you want to order against a customer agreement for an entire order. All order lines of the order will then use this agreement. See: Defining Agreements, *Oracle Order Entry/Shipping User's Guide*.

Order Holds and Credit Checking

Oracle Order Entry / Shipping automatically applies any appropriate, user–defined customer or item holds to an imported order. This lets you hold imported orders for review, just as you do orders entered through the Sales Orders window. Order Entry / Shipping also performs credit checking on all imported orders, according to the credit checking rules you define. See: Automatic Credit Checking of Orders, *Oracle Order Entry/Shipping User's Guide*.

Security Rules and Standard Value Rule Sets

OrderImport checks the security rules you have defined to ensure that any updates you make to imported orders are acceptable by your security standards.

Standard value rule sets work with OrderImport to default information to the fields in the Sales Orders window the same way as with orders entered directly. Standard value rule sets are assigned to the order type. See: Standard Value Rule Sets, *Oracle Order Entry/Shipping User's Guide*.

Internal Sales Orders

Oracle Purchasing uses OrderImport to transfer requisitions for internally sourced products to Order Entry / Shipping. Once imported, these internal sales orders are processed like regular sales orders. See: Creation of Internal Sales Orders, *Oracle Purchasing User's Guide*.

Drop Shipments

You can specify internal or external sourcing for the lines on an imported order by inserting data in the SOURCE_TYPE_CODE column in the SO_LINES_INTERFACE table. Once imported, these orders are processed like regular drop–ship or internally fulfilled orders entered in the Sales Orders window. See: Drop Shipments, *Oracle Order Entry/Shipping User's Guide*.

Notes

Oracle Order Entry / Shipping applies any automatic standard notes to imported orders that meet your automatic note criteria. Once an order is imported, you can edit these notes from the Sales Orders form as you would on a regular order. See: Defining Notes in Advance, *Oracle Order Entry/Shipping User's Guide.*

Reporting

Each time you run OrderImport, Order Entry / Shipping automatically generates the OrderImport Processing Results Report, which tells you the total number of orders imported successfully and any warnings about scheduling. Scheduling warnings include insufficient quantity on–hand to reserve an item. If an order line has validation errors, OrderImport rejects the entire order. Use the Request window to view this report, and indicate either the concurrent request ID or the program name.

Order Entry / Shipping reports all errors that occur during the import process in the OrderImport Process Exception Report. You can then correct any errors and successfully reimport your data.

See Also

Overview of Sales Orders, *Oracle Order Entry/Shipping User's Guide*

OrderImport Processing Results Report, *Oracle Order Entry/Shipping User's Guide*

OrderImport Process Exception Report, *Oracle Order Entry/Shipping User's Guide*

Understanding the Interface Tables

Order Entry / Shipping uses seven *Oracle* tables in which OrderImport receives data you import from other systems.

- SO_HEADERS_INTERFACE
- SO_HEADER_ATTRIBUTES_INTERFACE
- SO_LINES_INTERFACE
- SO_LINE_ATTRIBUTES_INTERFACE
- SO_LINE_DETAILS_INTERFACE
- SO_PRICE_ADJUSTMENTS_INTERFACE
- SO_SALES_CREDITS_INTERFACE

When OrderImport receives data, it validates and converts your import data into orders within Order Entry / Shipping. These interface tables are organized by columns in which Order Entry / Shipping categorizes and stores specific data. For example, your original order number information is stored in the column called ORIGINAL_SYSTEM_REFERENCE in the table SO_HEADERS. (SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE)

Prerequisites

Order Import expects that information such as customers, agreements, items, and salespersons will be defined before you interface records referencing them. To ensure that OrderImport works for you, you should prepare Oracle Order Entry / Shipping for any new data that you want OrderImport to import. Therefore, before importing orders, you should:

Define QuickCodes

Add QuickCodes to Oracle Order Entry / Shipping if your feeder systems use QuickCode values not yet defined in Oracle Order Entry / Shipping.

Define Sales Credit Types

Add sales credit types to Oracle Order Entry / Shipping if your feeder systems use sales credit types not yet defined.

Define Items

Add items to Inventory using the Master Item window if your feeder systems use items not yet defined.

Define Configurations

Define configurations in Oracle Order Entry / Shipping if your feeder systems use configurations not yet defined. You need first to define each model, option class, option item, standard component, and included item as an item. Then use the Bills of Material window to set up your configuration structures.

Define Price Lists

Add price lists to Oracle Order Entry / Shipping if your feeder systems use price lists not yet defined. If you import the list and selling price for your order lines, you do not need to add each item to a price list; you can simply reference a price list name that has no lines. However, you cannot add price adjustments to an order line without the item being on the price list.

Define Discounts

Add discounts to Oracle Order Entry / Shipping if your feeder systems use discounts not yet defined. If you specify that you want to calculate prices manually for imported orders (SO_LINES_INTERFACE.CALCULATE_PRICE is No), and you insert a list price that is different from the selling price, you also need to reference an existing discount to capture the difference. You can specify line–level discounts with the Automatic field set to No. The total difference between the list and selling price must be represented by price adjustments. Also, if you want to apply order level discounts to imported orders, you need to import the discount through the SO_PRICE_ADJUSTMENTS_INTERFACE table, because order level price adjustments are not automatically applicable.

Define Order Types

Add order types to Oracle Order Entry / Shipping if your feeder systems use order types not yet defined. If your order type references a standard value rule set, defaulting will automatically occur as your order is imported.

Define OrderImport Sources

Define OrderImport sources and determine whether to use IDs. See: Using IDs: page 5 – 55.

Enter Customers and Customer Sites

Add customers and customer addresses to Oracle Order Entry / Shipping if your feeder systems use customers not yet defined. If you are using a feeder system to add new customers, run the Customer Interface program in Oracle Receivables before running OrderImport.

Define Salespersons

Add salespersons to Oracle Order Entry / Shipping if your feeder systems use salespersons not yet defined.

Define Customer Agreements

Add customer agreements to Oracle Order Entry / Shipping if your feeder systems use customer agreements not yet defined.

Standard Value Rule Sets

Define standard value rule sets to default information to the fields in the Sales Orders window, based on other information you enter. Assign your standard value rule sets to order types.

See Also

Overview of Order Entry / Shipping Setup, *Oracle Order Entry/Shipping User's Guide*

Customer Interface, *Oracle Receivables User's Guide*

Defining Order Entry / Shipping QuickCodes, *Oracle Order Entry/Shipping User's Guide*

Defining Receivables QuickCodes, *Oracle Receivables User's Guide*

QuickCodes, *Oracle Applications System Administrator's Guide*

Defining Sales Credit Types, *Oracle Order Entry/Shipping User's Guide*

Overview of Pricing, *Oracle Order Entry/Shipping User's Guide*

Defining Price Lists, *Oracle Order Entry/Shipping User's Guide*

Defining Discounts, *Oracle Order Entry/Shipping User's Guide*

Defining Order Types, *Oracle Order Entry/Shipping User's Guide*

Defining OrderImport Sources, *Oracle Order Entry/Shipping User's Guide*

Defining Salespersons, *Oracle Receivables User's Guide*

Entering Customers, *Oracle Receivables User's Guide*

Defining Agreements, *Oracle Order Entry/Shipping User's Guide*

Defining Standard Value Rule Sets, *Oracle Order Entry/Shipping User's Guide*

Defining Items, *Oracle Inventory User's Guide*

Creating a Bill of Material, *Oracle Bills of Material User's Guide*

Importing Data From Your Feeder System

OrderImport receives data from your import program, which must convert data from your feeder system into a standard data format that OrderImport can read. OrderImport can then convert your import data into Order Entry / Shipping orders.

The type of environment from which you import data determines the type of import program you need, as well as the import utility you should use to write your program. SQL*Loader is a powerful and easy–to–use tool you can use to write such a program. However, depending on the complexity of your needs, there are a number of programming languages available to you. You can also use embedded 3GL Oracle tools like Pro*C, Pro*COBOL, and Pro*Fortran to write your program.

Identifying Source IDs

You can find the ORDER_SOURCE_ID in the SO_ORDER_SOURCES table for each OrderImport source that you define. Use the ORDER_SOURCE_ID when importing orders.

Assigning Values for System Items Flexfield Segments

Depending on the ORDER_SOURCE_ID you choose, the SO_ORDER_SOURCES.USE_IDS flag is either **Y** or **N**. If the USE_IDS flag is Y, the ID columns are used. If the USE_IDS flag is N, NAME or SEGMENT columns are used.

If your USE_IDS flag is set to **N**, assign a System Items flexfield value to as many segments as you have enabled. As long as you enter the segments in the correct sequence, OrderImport knows which segment columns to place the values in. For example, if you enabled six Item flexfield segments in Oracle Inventory, you can assign values to columns SEGMENT1 through SEGMENT6. If you enabled only four Item flexfield segments, then you can assign values to columns SEGMENT1 through SEGMENT4.

MTL_SYSTEM_ITEMS column	Flexfield Sequence	SO_LINES_INTERFACE column
SEGMENT1	1	INVENTORY_ITEM_SEGMENT1
SEGMENT20	2	INVENTORY_ITEM_SEGMENT2
SEGMENT13	3	INVENTORY_ITEM_SEGMENT3
SEGMENT5	4	INVENTORY_ITEM_SEGMENT4
SEGMENT17	5	INVENTORY_ITEM_SEGMENT5
SEGMENT3	6	INVENTORY_ITEM_SEGMENT6

Table 5 – 5 Flexfield Segment Correspondences (Page 1 of 1)

Load valid enabled segment values for your enabled segments into the OrderImport tables. The segment values must already be defined. Made sure you specify the segment value correctly. For example, the value **01** is not the same as the value **1**. OrderImport does not allow null values in enabled segments. See: Oracle Inventory Flexfields, *Oracle Inventory User's Guide*.

Assigning Values to Not Null Columns

You must assign values to all Not Null columns in the OrderImport tables in order for OrderImport to convert your import data successfully into orders.

Assigning Values to Optional Columns

You can enter optional column values for several columns in the OrderImport tables. Entering values in these optional columns gives you maximum control over the way OrderImport handles your data.

Importing Configurations

OrderImport lets you easily import models or configurations, option classes, and option items. Standard components and included items are not imported, but Order Entry / Shipping derives them from the associated bill of material (BOM). Only Order Entry / Shipping can explode a bill of material. The bill of material structure for an order line (SO_LINES_INTERFACE.ORIGINAL_SYSTEM_LINE_ REFERENCE) is determined by the combination of PARENT_LINE_REF, LINK_TO_LINE_REF, ITEM_TYPE_CODE, and OPTION_FLAG columns in the SO_LINES_INTERFACE table.

The following table shows an example of a configuration that is four levels deep with option classes and models embedded at different levels, and how to use the columns listed above to explain the bill structure to Order Entry / Shipping. See: Overview of Bills of Material, *Oracle Bills of Material User's Guide.*

Level		ORIGINAL_SYSTEM_ LINE_REFERENCE	PARENT_LINE_REF	LINK_TO_LINE_REF	ITEM_TYPE_CODE	OPTION_FLAG
1	Model (top level)	1			Model	
2	.Class A	2	1	1	Option Class	Y
3	..Class A Option	3	1	2	Standard	Y
2	.Model B	4	1	1	Model	Y
3	..Model B Option	5	1	4	Standard	Y
3	..Class C	6	1	4	Option Class	Y
4	...Class C Option	7	1	6	Kit	Y

Table 5 – 6 Sample Bill Structure (Page 1 of 1)

Importing Schedule Details

You can import schedule details for an item on an order line. When you import a schedule detail, you must specify the line to which it applies, and you can only specify details for the same inventory item that is on the order line.

Importing Customer–requested Shipment Schedules

Imported, customer–requested shipment schedules must be separate order lines, unlike in on–line order entry, where shipment schedules are related to an order line. If you are creating shipment schedules to gain

price advantages, you will need to import the appropriate price for each line. If you use automatic pricing, each order line (imported shipment schedule) is evaluated separately and volume discounts across lines are not considered. See: Overview of Order Scheduling, *Oracle Order Entry/Shipping User's Guide.*

Managing Customer Requests to Ship Orders Completely

OrderImport provides a line–level field to control whether the lines on a order can ship independently or must ship together. If this field, SHIP_SET_NUMBER, is populated with the same number for each line, the entire order will have to be ready before it will be released for picking.

Using MultiCurrency and Price Lists

You must indicate the currency of each order you import. If you have price lists in different currencies, you can load both the price list and matching currency through OrderImport. Alternatively, you can default the price list according to the standard value rules you have defined. If the defaulted price list does not have the same currency as what you imported, Order Entry / Shipping rejects the transaction. See: Defining Standard Value Rule Sets, *Oracle Order Entry/Shipping User's Guide.*

Changes to Imported Orders

You can import changes to imported orders. When entering changes to orders you need to identify the order, order line(s), price adjustments, and sales credits, and match the columns in the SO_HEADERS table to the corresponding ones in the SO_HEADERS_INTERFACE table, shown below.

Specific Record to Change	Match This Order Entry Column...	...to This OrderImport Interface Column
Header	**SO_HEADERS columns**	**SO_HEADERS_INTERFACE columns**
	ORIGINAL_SYSTEM_SOURCE_CODE	ORDER_SOURCE_ID
	ORIGINAL_SYSTEM_REFERENCE	ORIGINAL_SYSTEM_REFERENCE
Line	**SO_HEADERS columns**	**SO_LINES_INTERFACE columns**
	ORIGINAL_SYSTEM_SOURCE_CODE	ORDER_SOURCE_ID
	ORIGINAL_SYSTEM_REFERENCE	ORIGINAL_SYSTEM_REFERENCE
	SO_LINES column	**SO_LINES_INTERFACE column**
	ORIGINAL_SYSTEM_LINE_REFERENCE	ORIGINAL_SYSTEM_LINE_REFERENCE
Price Adjustments	**SO_HEADERS columns**	**SO_PRICE_ADJUSTMENTS_INTERFACE columns**
	ORIGINAL_SYSTEM_SOURCE_CODE	ORDER_SOURCE_ID
	ORIGINAL_SYSTEM_REFERENCE	ORIGINAL_SYSTEM_REFERENCE
	SO_LINES column	**SO_PRICE_ADJUSTMENTS_INTERFACE column**
	ORIGINAL_SYSTEM_LINE_REFERENCE	ORIGINAL_SYSTEM_LINE_REFERENCE
Sales Credits	**SO_HEADERS columns**	**SO_SALES_CREDITS_INTERFACE columns**
	ORIGINAL_SYSTEM_SOURCE_CODE	ORDER_SOURCE_ID
	ORIGINAL_SYSTEM_REFERENCE	ORIGINAL_SYSTEM_REFERENCE
	SO_LINES column	**SO_SALES_CREDITS_INTERFACE column**
	ORIGINAL_SYSTEM_LINE_REFERENCE	ORIGINAL_SYSTEM_LINE_REFERENCE

Table 5 – 7 Header– and Line–level Column Correspondences (Page 1 of 1)

You need to specify the operation code to determine the kind of change you want to import. Use the OPERATION_CODE column in the interface tables (except for SO_LINE_DETAILS_INTERFACE) to indicate whether you are adding, modifying or deleting imported order information.

When you want to create a new imported order, enter INSERT or null in SO_HEADERS_INTERFACE.OPERATION_CODE. When you want

to delete an imported order, enter DELETE in SO_HEADERS_INTERFACE.OPERATION_CODE.

When you want to update an order, you must reenter all the required and conditionally required columns for the order along with any changes. When you create new order lines, order price adjustments, order line price adjustments, order sales credits, or order line sales credits for an order you have already imported, you must update the order header. For line–level additions, you must also update the appropriate order line(s).

When you want to create or delete an order line, order price adjustment, or order sales credit for an order you have already imported, enter UPDATE in SO_HEADERS_INTERFACE. OPERATION_CODE (and reenter all the required columns), and INSERT or DELETE in the appropriate OPERATION_CODE column in any of SO_LINES_INTERFACE, SO_PRICE_ADJUSTMENTS_ INTERFACE, or SO_SALES_CREDITS_INTERFACE. If your order includes a discount at either the order or line level, you must reenter a price list (in SO_HEADERS_INTERFACE.PRICE_LIST_NAME if you use names, or SO_HEADERS_INTERFACE.PRICE_LIST_ID if you use IDs).

When you want to create or delete an order line price adjustment or order line sales credit for an order line you have already imported, you must enter UPDATE in SO_HEADERS_INTERFACE. OPERATION_CODE (and reenter all the required columns), UPDATE in SO_LINES_INTERFACE.OPERATION_CODE (and reenter all the required columns), and INSERT or DELETE in the appropriate OPERATION_CODE column of SO_PRICE_ADJUSTMENTS_ INTERFACE, or SO_SALES_CREDITS_INTERFACE.

When you want to delete an order line for a model, enter UPDATE in SO_HEADERS_INTERFACE.OPERATION_CODE (and reenter all the required columns), and DELETE in the appropriate OPERATION_CODE column of SO_LINES_INTERFACE for the model. This action deletes both the model line and all its associated option classes, option items, and included items.

When you want to update an order line for a model that you have already imported, enter UPDATE in SO_HEADERS_INTERFACE. OPERATION_CODE (and reenter all the required columns), and DELETE and INSERT in the appropriate OPERATION_CODE column of any of SO_LINES_INTERFACE, SO_PRICE_ADJUSTMENTS_ INTERFACE, or SO_SALES_CREDITS_INTERFACE for the model and all its associated option classes, option items, and included items. You

must give the inserting model a new line number since you are simultaneously deleting and inserting in order to update the line.

When you want to update an order that has multiple order lines, some reserved or demanded and some not, you can update the unscheduled order line(s) without affecting the schedule status of the reserved or demanded lines. Enter UPDATE in SO_HEADERS_INTERFACE. OPERATION_CODE (and reenter all the required columns), and set SO_LINES_INTERFACE.OPERATION_CODE to UPDATE for the unscheduled order lines you want to modify. You cannot import changes to an order line that is reserved or demanded. Instead, query the order in the Sales Orders or Schedule Orders windows and modify the corresponding field(s).

You can insert a new order line into an existing schedule group (ship set, ATO configuration, or PTO Ship Model Complete configuration), until the schedule group is demanded or reserved.

When you want to update an order line that is part of a schedule group (ship set, ATO configuration, or PTO Ship Model Complete configuration), you can modify Ordered Quantity and Schedule Date prior to demanding or reserving the schedule group. The following columns cannot be changed at any time through OrderImport for lines in a schedule group: Warehouse, Ship–to Location, Freight Carrier, Shipment Priority, and Demand Class. Instead, query the order in the Sales Orders or Schedule Orders windows and modify schedule group attributes.

When you import changes to any schedule details (SO_LINE_DETAILS_INTERFACE), you need to reimport *all* the schedule details for the order line. Enter UPDATE in SO_HEADERS_INTERFACE.OPERATION_CODE (and reenter all the required columns), and set SO_LINES_INTERFACE. OPERATION_CODE to UPDATE for the corresponding order line. OrderImport deletes existing schedule details for the order line and replaces those with the newly imported schedule detail information. The new schedule details totals must add up to the total quantity for the order line. If an order line is reserved or demanded, any changes to schedule details through OrderImport will result in the order line and its schedule details being unscheduled. When changing schedule details, you can reschedule the entire *order* by setting SO_HEADERS_INTERFACE.SCHEDULE_STATUS_CODE to RESERVED or DEMANDED.

Attention: If you want to delete a column, you cannot do so by updating the interface table. You should query the order in

the Orders Workbench and delete the data from the corresponding field.

The operation codes that you need to indicate for each kind of changes are outlined in the following table. Operation codes in angle brackets (< >) are optional; all others are required for the operation.

Operation	SO_HEADERS_INTERFACE	SO_LINES_INTERFACE	SO_PRICE_ADJUSTMENTS_ INTERFACE	SO_SALES_CREDITS_INTERFACE
Add order	<Insert>	<Insert>	<Insert>	<Insert>
Add order line	Update	Insert	<Insert>	<Insert>
Delete order	Delete	Delete	<Delete>	Delete
Delete order line	Update	Delete	<Delete>	<Delete>
Update order	Update	—	—	—
Update order line	Update	Update	<Update>	<Update>
Update order price adjustment	Update	—	Update	—
Update order sales credit	Update	—	—	Update
Update line price adjustment	Update	Update	Update	—
Update line sales credit	Update	Update	—	Update
Delete model	Update	Delete		
Update model	Update	Delete *and* Insert	<Insert>	<Insert>

Table 5 – 8 Operation Codes (Page 1 of 1)

Update Statements

The following update statements are necessary to reprocess orders in the OrderImport interface tables. First fix the problems/errors in the orders and ensure they are ready for OrderImport to process them again. Then use the following statement in SQL*Plus for each interface table to remove values from the necessary columns.

Attention: Values or IDs specific to your situation appear in italics within angle brackets (< >). Be prepared to substitute the specific value or ID for the generic value in the angle

brackets. Do not include the angle brackets when substituting your values for these names.

```
update so_headers_interface
set request_id = NULL, error_flag = NULL,
interface_status = NULL, report_summary = NULL
where
<insert where clause here>
```

The where clause could include statements like

- `where request_id = <xxxxx>`
- `where order_source_id = <yyyy>`
- `where original_system_reference in ('<aaa>','<bbb>')`

or any combination of the above.

Using IDs

You can define sources and determine whether to use IDs in the OrderImport Sources window. If you elect to use IDs, you must populate all required ID columns in each table. You may also populate the associated name columns; however, Oracle Order Entry / Shipping does not consider names when you decide to use IDs.

If you elect *not* to use IDs, then you must populate all the required name columns in each table. You may also populate the ID columns; however, except for those columns listed below, Oracle Order Entry / Shipping does not consider the IDs when you decide to use names.

If you import orders using names and you have a nonunique customer, address information, contact, or salesperson, you can include the IDs for only the columns indicated in the following table. OrderImport uses the IDs along with the names to identify the specific customer, address or salesperson. The following table shows the corresponding name and ID columns.

NAME COLUMN	ID COLUMN
SO_HEADERS_INTERFACE	
CUSTOMER_NAME CUSTOMER_NUMBER	CUSTOMER_ID
ORDERED_BY_CONTACT_FIRST_NAME ORDERED_BY_CONTACT_LAST_NAME	ORDERED_BY_CONTACT_ID
INVOICE_CUSTOMER (name)	INVOICE_CUSTOMER_ID
INVOICE_ADDRESS1 INVOICE_ADDRESS2 INVOICE_ADDRESS3 INVOICE_ADDRESS4 INVOICE_CITY INVOICE_STATE INVOICE_COUNTY INVOICE_COUNTRY INVOICE_POSTAL_CODE	INVOICE_ADDRESS_ID
INVOICE_TO_CONTACT_FIRST_NAME INVOICE_TO_CONTACT_LAST_NAME	INVOICE_TO_CONTACT_ID
SHIP_TO_CUSTOMER (name)	SHIP_TO_CUSTOMER_ID
SHIP_ADDRESS1 SHIP_ADDRESS2 SHIP_ADDRESS3 SHIP_ADDRESS4 SHIP_CITY SHIP_STATE SHIP_COUNTY SHIP_COUNTRY SHIP_POSTAL_CODE	SHIP_ADDRESS_ID
SHIP_TO_CONTACT_FIRST_NAME SHIP_TO_CONTACT_LAST_NAME	SHIP_TO_CONTACT_ID
SALESREP_NAME	SALESREP_ID
SALESREP_NUMBER	

Table 5 – 9 Name and ID Correspondences (Page 1 of 2)

NAME COLUMN	ID COLUMN
SO_LINES_INTERFACE	
SHIP_TO_CUSTOMER (name)	SHIP_TO_CUSTOMER_ID
SHIP_ADDRESS1 SHIP_ADDRESS2 SHIP_ADDRESS3 SHIP_ADDRESS4 SHIP_CITY SHIP_STATE SHIP_COUNTY SHIP_COUNTRY SHIP_POSTAL_CODE	SHIP_ADDRESS_ID
SHIP_TO_CONTACT_FIRST_NAME SHIP_TO_CONTACT_LAST_NAME	SHIP_TO_CONTACT_ID

Table 5 – 9 Name and ID Correspondences (Page 2 of 2)

If the name, or number, and ID combination you supply does not match an existing customer, address, or salesperson, OrderImport will reject the records. (For the INVOICE_CUSTOMER and SHIP_TO_CUSTOMER columns, you must enter the name; you cannot use a customer number.) See: Defining OrderImport Sources, *Oracle Order Entry/Shipping User's Guide.*

Standard Value Rule Sets

In addition to entering data in the columns of the interface tables, you can utilize the standard value defaulting feature to fill in values to each imported order. You specify a standard value rule set with each order type. Any information that you enter in the columns of the interface tables, which can also be defaulted by the standard value rule set, will be treated as values that a user entered. Therefore, you should set the override fields of the Standard Value Rule Sets window according to how you want Oracle Order Entry / Shipping to prioritize user–entered values relative to defaulted values. See: Defining Standard Value Rule Sets, *Oracle Order Entry/Shipping User's Guide.*

OrderImport Validation

OrderImport validates your data for compatibility with Oracle Order Entry / Shipping. OrderImport validates your data by ensuring that the columns in the OrderImport interface tables reference the appropriate and active values and columns in Oracle Order Entry / Shipping.

Whenever IDs are used, OrderImport automatically validates the name and ID combination, as well as the individual name and ID.

If you are importing open orders, then for those attributes that must be valid, the following rules apply: if an attribute has start and end dates (such as agreements or price lists), OrderImport validates that the end date is greater than the current date; if the attribute has an enable flag (such as QuickCodes), OrderImport validates that the flag is set to enabled; if the attribute has a status (such as customer), OrderImport validates that the status is Active.

If you are importing closed orders, then for those attributes that need only to exist, the start and end dates, enable flag, and status are irrelevant.

Transaction–level Validation

OrderImport validates the following attributes to ensure that your transactions contain the appropriate valid information for Oracle Order Entry/Shipping:

- customer
- ship–to and bill–to locations/addresses
- customer agreement
- contacts
- order type
- OrderImport source
- order category
- entry status
- salesperson
- price list
- currency
  - conversion type
- payment terms
- payment type
  - credit card type
- sales channel
- shipment priority, FOB, freight carrier, and freight terms

- accounting rule
- invoicing rule

Transaction Line–level Validation

OrderImport validates the following attributes to ensure that your transaction lines contain the appropriate valid information for Oracle Order Entry / Shipping:

- unit of measure
- inventory item
- ship–to address
- freight carrier
- discount (price adjustment)
- warehouse
- configuration (against bill of material for the top model)

Complete Orders Validation

Since you should be importing complete orders only for historical purposes, OrderImport relaxes validation on this type of order. Instead of requiring that attributes be valid as described above, OrderImport requires only that they exist in your database. For inventory items, the *Customer Ordered Item* attribute must be set to Yes; the *Customer Orders Enabled* attribute can be set to either Yes or No.

- existing customer
- existing ship–to and bill–to addresses
- existing and orderable inventory item
- existing price list
- existing salesperson

Security Rules and Standard Value Rule Sets Validation

OrderImport checks security rule and standard value rule sets after every order is inserted or updated in the interface tables. If you try to update an order at a point in its order cycle that violates your security rules, OrderImport yields an error. Standard value rule sets are reevaluated after every update to see if any defaulted values should be changed according to your rule.

Sales Credit Validation

OrderImport validates the following attributes to ensure that your sales credit information contains the appropriate valid information for Oracle Order Entry / Shipping:

- sales credit type
- salesperson
- total revenue percent for the order or each line is not greater than 100

Price Adjustment Validation

OrderImport validates the following attributes to ensure that your price adjustments contain the appropriate valid information for Oracle Order Entry / Shipping:

- discount (price adjustment)
- price list

Schedule Details Validation

OrderImport validates the following attributes to ensure that your schedule details contain the appropriate valid information for Oracle Order Entry / Shipping:

- subinventory
- lot
- revision
- total quantity of schedule details equals total ordered quantity on associated order line

Required Fields for Booked Orders

If you are importing booked orders, verify either that you have entered all fields required to book the order or that your standard value rules will default them. If all the required fields are not completed, OrderImport will reject the records. At the beginning of each table description, we provide tables that indicate columns that are conditionally required for booking. See: Required Fields for Entering Orders, *Oracle Order Entry/Shipping User's Guide.*

Running OrderImport

Running OrderImport can be a one–step process, depending on the accuracy of your import program. If your import program converts your transaction information from other sources into the required format, and all your data passes the validation in Oracle Order Entry / Shipping, then you should be able to run OrderImport successfully in one execution. However, if you load data into the OrderImport interface tables that Order Entry / Shipping validation rejects, OrderImport informs you of the specific validation errors in the concurrent request report. In this case, you need to correct any errors and reimport the corrected data. Before you reimport the corrected data, set the REQUEST_ID, ERROR_FLAG, and INTERFACE_STATUS values to null in all interface tables. Otherwise, OrderImport will not process any records. You can import one or multiple sources of data at one time, based on the parameters you set for the OrderImport program when you submit the program request. See: OrderImport, *Oracle Order Entry/Shipping User's Guide*.

Oracle Order Entry/Shipping Interface Tables and Column Descriptions

OrderImport uses the following tables and columns.

SO_HEADERS_INTERFACE

The following summarizes the SO_HEADERS_INTERFACE table. Database columns with a check under Conditionally Required for Booking are optional if you are importing an order with any valid status other than **Booked**. If you are importing an order with an Entry Status of **Booked**, you must enter the ID column if you are using IDs, or the name column if you are not using IDs. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 66.

SO_HEADERS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number				✓
PROGRAM_ APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_UPDATE_ DATE	Date				✓
REQUEST_ID	Number				✓
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
ORDER_NUMBER_ SOURCE_ID	Number			✓	
CUSTOMER_NAME	Varchar2(50)		✓		
CUSTOMER_NUMBER	Varchar2(30)		✓		
CUSTOMER_ID	Number		✓		
ORDER_TYPE	Varchar2(30)	**C**			
ORDER_TYPE_ID	Number	**C**			
ORDER_SOURCE_ID	Number	✓			
ORDER_CATEGORY	Varchar2(30)	✓			
DATE_ORDERED	Date	✓			
CURRENCY_CODE	Varchar2(15)	✓			
CONVERSION_RATE	Number	**C**			
CONVERSION_DATE	Date	**C**			

Table 5 – 10 SO_HEADERS_INTERFACE (Page 1 of 5)

SO_HEADERS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
CONVERSION_ TYPE_CODE	Varchar2(30)	C			
SALESREP_NAME	Varchar2(30)		✓		
SALESREP_ID	Number		✓		
INVOICE_CUSTOMER	Varchar2(50)		✓		
INVOICE_CUSTOMER_ID	Number		✓		
INVOICE_TO_ SITE_USE_ID	Number			✓	
INVOICE_ADDRESS_ID	Number		✓		
INVOICE_ADDRESS1–4	Varchar2(240)		✓		
INVOICE_CITY	Varchar2(50)		✓		
INVOICE_STATE	Varchar2(50)		✓		
INVOICE_COUNTY	Varchar2(50)		✓		
INVOICE_COUNTRY	Varchar2(50)		✓		
INVOICE_POSTAL_CODE	Varchar2(30)		✓		
SHIP_TO_SITE_USE_ID	Number			✓	
SHIP_TO_CUSTOMER	Varchar2(50)		✓		
SHIP_TO_CUSTOMER_ID	Number		✓		
SHIP_ADDRESS_ID	Number		✓		
SHIP_ADDRESS1–4	Varchar2(240)		✓		
SHIP_CITY	Varchar2(50)		✓		
SHIP_STATE	Varchar2(50)		✓		
SHIP_COUNTY	Varchar2(50)		✓		
SHIP_COUNTRY	Varchar2(50)		✓		
SHIP_POSTAL_CODE	Varchar2(30)		✓		
PRICE_LIST_NAME	Varchar2(30)		✓		

Table 5 – 10 SO_HEADERS_INTERFACE (Page 2 of 5)

SO_HEADERS_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
PRICE_LIST_ID	Number		✓		
TERMS_NAME	Varchar2(15)		✓		
TERMS_ID	Number		✓		
CYCLE_ID	Number			✓	
ENTERED_STATE_NAME	Varchar2(30)	C			
ENTERED_STATE_ID	Number	C			
ENTERED_STATE_DATE	Date				✓
COMPLETE_FLAG	Varchar2(1)				✓
AGREEMENT_NAME	Varchar2(30)				✓
AGREEMENT_ID	Number				✓
DATE_REQUESTED_CURRENT	Date				✓
ORDERED_BY_CONTACT_FIRST_NAME	Varchar2(30)				✓
ORDERED_BY_CONTACT_LAST_NAME	Varchar2(50)				✓
ORDERED_BY_CONTACT_ID	Number				✓
SALES_CHANNEL_CODE	Varchar2(30)				✓
PURCHASE_ORDER_NUM	Varchar2(50)				✓
INVOICE_TO_CONTACT_FIRST_NAME	Varchar2(30)				✓
INVOICE_TO_CONTACT_LAST_NAME	Varchar2(50)				✓
INVOICE_TO_CONTACT_ID	Number				✓
SHIP_TO_CONTACT_FIRST_NAME	Varchar2(30)				✓

Table 5 – 10 SO_HEADERS_INTERFACE (Page 3 of 5)

SO_HEADERS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
SHIP_TO_ CONTACT_LAST_NAME	Varchar2(50)				✓
SHIP_TO_CONTACT_ID	Number				✓
TAX_EXEMPT_NUM	Varchar2(30)				✓
TAX_EXEMPT_ REASON_CODE	Varchar2(30)	C			
TAX_EXEMPT_FLAG	Varchar2(1)				✓
SHIPMENT_ PRIORITY_CODE	Varchar2(30)				✓
SHIP_METHOD_CODE	Varchar2(30)				✓
FREIGHT_TERMS_CODE	Varchar2(30)				✓
FOB_CODE	Varchar2(30)				✓
SHIPPING_ INSTRUCTIONS	Varchar2(240)				✓
PACKING_INSTRUCTIONS	Varchar2(240)				✓
DATE_SHIPPED	Date				✓
PAYMENT_TYPE_CODE	Varchar2(30)				✓
PAYMENT_AMOUNT	Number				✓
CHECK_NUMBER	Varchar2(50)				✓
CREDIT_CARD_CODE	Varchar2(30)				✓
CREDIT_CARD_ HOLDER_NAME	Varchar2(50)				✓
CREDIT_CARD_NUMBER	Varchar2(50)				✓
CREDIT_CARD_ EXPIRATION_DATE	Date				✓
CREDIT_CARD_ APPROVAL_CODE	Varchar2(50)				✓
ACCOUNTING_RULE	Varchar2(30)		✓		

Table 5 – 10 SO_HEADERS_INTERFACE (Page 4 of 5)

SO_HEADERS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
ACCOUNTING_RULE_ID	Number		✓		
INVOICING_RULE	Varchar2(30)		✓		
INVOICING_RULE_ID	Number		✓		
CONTEXT	Varchar2(30)				✓
ATTRIBUTE1–15	Varchar2(150)				✓
SCHEDULE_ STATUS_CODE	Varchar2(30)				✓
OPERATION_CODE	Varchar2(30)				✓
SALESREP_NUMBER	Number		✓		✓
APPLY_STANDARD_ NOTES	Varchar2(1)				✓
DEMAND_CLASS_CODE	Varchar2(30)				
ORG_ID	Number				✓

Table 5 – 10 SO_HEADERS_INTERFACE (Page 5 of 5)

CREATION_DATE Not Null **DATE.**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_HEADERS.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created the record.

Validation: None

Destination: SO_HEADERS.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the current date on which you are entering data into your feeder system.

Validation: Standard date validation

Destination: SO_HEADERS.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created or who most recently modified the record.

Validation: None

Destination: SO_HEADERS.LAST_UPDATED_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Order Entry / Shipping with this data.

Validation: None

Destination: SO_HEADERS.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER(15)**

This column is for internal use only.

PROGRAM_ID **NUMBER(15)**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER(15)**

This column is for internal use only. It is populated with the concurrent manager request ID for the run of the interface each time you submit the OrderImport program. In order for records to be processed, their request ID must be null or equal to the request ID being processed.

ORIGINAL_SYSTEM_REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from your original feeder system. The order number of ID you enter provides you with an audit trail from OrderImport to your feeder system.

Validation: If the value you enter already exists, then the OPERATION_CODE must be **UPDATE** or **DELETE**.

Destination: SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE (This order number appears in the Sales Orders and View Orders windows.)

ORDER_NUMBER_SOURCE_ID **NUMBER**

The information for this column is derived from the order type and used to generate the order number. It contains the ID of the order number source to be used when generating the order number from SO_ORDER_NUMBER_SOURCES.

Validation: None

Destination: None

Defining Order Types, *Oracle Order Entry/Shipping User's Guide*

Defining Order Number Sources, *Oracle Order Entry/Shipping User's Guide*

CUSTOMER_NAME **VARCHAR2(50)**

Enter the full name of the customer for this order. If you are using IDs, you must enter the CUSTOMER_ID. If you are not using IDs, then you must enter the CUSTOMER_NAME or CUSTOMER_NUMBER, and if your CUSTOMER_NAME and/or CUSTOMER_NUMBER is not unique, you must also enter the CUSTOMER_ID.

Validation: RA_CUSTOMERS.CUSTOMER_NAME

Destination: inserts RA_CUSTOMERS.CUSTOMER_ID into SO_HEADERS.CUSTOMER_ID where SO_HEADERS_INTERFACE.CUSTOMER_NAME = RA_CUSTOMERS.CUSTOMER_NAME

CUSTOMER_NUMBER **VARCHAR2(30)**

Enter the customer number for this order. If you are using IDs, you must enter the CUSTOMER_ID. If you are not using IDs, then you must enter the CUSTOMER_NAME or CUSTOMER_NUMBER, and if your CUSTOMER_NAME and/or CUSTOMER_NUMBER is not unique, you must also enter the CUSTOMER_ID.

Validation: RA_CUSTOMERS.CUSTOMER_NUMBER

Destination: None

CUSTOMER_ID **NUMBER**

Enter the customer ID that matches your customer name. If you are using IDs, you must enter the CUSTOMER_ID. If you are not using IDs, then you must enter the CUSTOMER_NAME or CUSTOMER_NUMBER, and if your CUSTOMER_NAME and/or CUSTOMER_NUMBER is not unique, you must also enter the CUSTOMER_ID.

Validation: RA_CUSTOMERS.CUSTOMER_ID

Destination: SO_HEADERS.CUSTOMER_ID

ORDER_TYPE Conditionally Not Null **VARCHAR2(30)**

Enter the name of the order type to assign to your order. If you are using IDs, you must enter the ORDER_TYPE_ID. If you are not using IDs, then you must enter the ORDER_TYPE_NAME.

Validation: SO_ORDER_TYPES.NAME

Destination: inserts SO_ORDER_TYPES.ORDER_TYPE_ID into SO_HEADERS.ORDER_TYPE_ID where SO_HEADERS_INTERFACE.ORDER_TYPE = SO_ORDER_TYPES.NAME

ORDER_TYPE_ID Conditionally Not Null **NUMBER**

Enter the order type ID that matches your order type. If you are using IDs, you must enter the ORDER_TYPE_ID. If you are not using IDs, then you must enter the ORDER_TYPE_NAME.

Validation: SO_ORDER_TYPES.ORDER_TYPE_ID

Destination: SO_HEADERS.ORDER_TYPE_ID

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID. This tells OrderImport whether to use IDs or the name columns, and is used to determine which records in the interface table to process. See: Defining OrderImport Sources, *Oracle Order Entry/Shipping User's Guide.*

Validation: SO_ORDER_SOURCES.ORDER_SOURCE_ID

Destination: None

ORDER_CATEGORY Not Null **VARCHAR2(30)**

Enter **R** in this column. This stands for a Regular order.

Validation: Must equal **R**

Destination: SO_HEADERS.ORDER_CATEGORY

DATE_ORDERED Not Null **DATE**

Enter the date on which this order was placed in the original system.

Validation: Standard date validation

Destination: SO_HEADERS.DATE_ORDERED

CURRENCY_CODE Not Null **VARCHAR2(15)**

Enter the currency for the order.

Validation: FND_CURRENCIES

Destination: SO_HEADERS.CURRENCY_CODE

CONVERSION_RATE Conditionally Not Null **NUMBER**

Enter the currency conversion rate for the order if you entered a currency code other than the functional currency for your set of books and a conversion_type_code equal to **User**.

Validation: None

Destination: SO_HEADERS.CONVERSION_RATE

CONVERSION_DATE Conditionally Not Null **DATE**

Enter the currency conversion date for which the conversion rate is valid for the order, if you entered a currency code other than the

functional currency for your set of books and a conversion_type_code equal to **User**.

Validation: None

Destination: SO_HEADERS.CONVERSION_DATE

CONVERSION_TYPE_CODE Conditionally Not Null **VARCHAR2(30)**

Enter the currency conversion type for the order if you entered a currency code other than the functional currency for your set of books.

Validation: GL_DAILY_CONVERSION_TYPES. CONVERSION_TYPE

Destination: SO_HEADERS.CONVERSION_TYPE_CODE

SALESREP_NAME **VARCHAR2(30)**

Enter the name of the primary salesperson for this order. (If you do not enter explicit sales credit records, OrderImport will automatically assign 100% quota credit for the whole order to this salesperson). If you are using IDs, you must enter the SALESREP_ID. If you are not using IDs, then you must enter the SALESREP_NAME or SALESREP_NUMBER, and if your SALESREP_NAME and/or SALESREP_NUMBER is not unique, you must also enter the SALESREP_ID. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_SALESREPS.NAME

Destinations: inserts RA_SALESREPS.SALESREP_ID into SO_HEADERS.SALESREP_ID where RA_SALESREP.NAME = SO_HEADERS_INTERFACE.SALESREP_NAME

SALESREP_ID **NUMBER**

Enter the salesrep ID for the salesperson. If you are using IDs, you must enter the SALESREP_ID. If you are not using IDs, then you must enter the SALESREP_NAME or SALESREP_NUMBER, and if your SALESREP_NAME and/or SALESREP_NUMBER is not unique, you must also enter the SALESREP_ID. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_SALESREPS.SALESREP_ID

Destination: SO_HEADERS.SALESREP_ID

INVOICE_CUSTOMER **VARCHAR2(50)**

Enter the name of the customer to receive the invoice for this order. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Validates whether the INVOICE_CUSTOMER exists in RA_CUSTOMERS and RA_CUSTOMERS.STATUS is **Active**. (If the customer name is not unique, you must enter an ID in the INVOICE_CUSTOMER_ID column.) Also, if *OE: Customer Relationships* is set to Yes, then the bill–to location must match the ordering customer or a related customer.

Destination: None

INVOICE_CUSTOMER_ID Conditionally Not Null **NUMBER**

Enter the customer ID of your INVOICE_CUSTOMER. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Validates whether the INVOICE_CUSTOMER_ID exists in RA_CUSTOMERS and RA_CUSTOMERS.STATUS is **Active**. Also, if *OE: Customer Relationships* is set to Yes, then the bill–to location must match the ordering customer or a related customer.

Destination: None

INVOICE_TO_SITE_USE_ID **NUMBER**

Leave this column null; it is derived from the INVOICE_ADDRESS_ID column.

Destination: inserts RA_SITE_USE.SITE_USE_ID into SO_HEADERS.INVOICE_SITE_USE_ID where RA_ADDRESSES.ADDRESS_ID = RA_SITE_USES.ADDRESS_ID and RA_SITE_USES.SITE_USE_CODE = 'BILL_TO'

INVOICE_ADDRESS_ID **NUMBER**

Enter the ADDRESS_ID which matches your complete bill–to address location. This value is used to derive SO_LINES_INTERFACE.INVOICE_TO_SITE_USE_ID. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Must represent a valid location for the INVOICE_CUSTOMER and exist in RA_SITE_USES where SITE_USE_CODE is **BILL_TO**. Validates whether the address is valid and exists in RA_ADDRESSES; must be a valid address for the INVOICE_CUSTOMER. Also, if *OE: Customer Relationships* is set to Yes, then the bill–to location must match the ordering customer or a related customer.

INVOICE_ADDRESS1–4 **VARCHAR2(240)**

Enter the street address (4 lines) for your bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

INVOICE_CITY **VARCHAR2(50)**

Enter the city for your bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

INVOICE_STATE **VARCHAR2(50)**

Enter the state or province for your bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

INVOICE_COUNTY **VARCHAR2(50)**

Enter the county for your bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

INVOICE_COUNTRY **VARCHAR2(50)**

Enter the country for your bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: FND_TERRITORIES

INVOICE_POSTAL_CODE **VARCHAR2(30)**

Enter the postal code for your bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_TO_SITE_USE_ID **NUMBER**

Leave this column null; it is derived from the SHIP_ADDRESS_ID column.

Destination: inserts RA_SITE_USE.SITE_USE_ID into SO_HEADERS.SHIP_TO_SITE_USE_ID where RA_ADDRESSES.ADDRESS_ID = RA_SITE_USES.ADDRESS_ID and RA_SITE_USES.SITE_USE_CODE = 'SHIP_TO'

SHIP_TO_CUSTOMER **VARCHAR2(50)**

Enter the name of the customer to receive the shipment. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Validates whether the SHIP_TO_CUSTOMER exists in RA_CUSTOMERS and RA_CUSTOMERS.STATUS is **Active**. (If the customer name is not unique, you must enter an ID in the SHIP_TO_CUSTOMER_ID column.) Also, if *OE: Customer Relationships* is set to Yes, then the bill–to location must match the ordering customer or a related customer.

Destination: None

SHIP_TO_CUSTOMER_ID Conditionally Not Null **NUMBER**

Enter the customer ID of your ship–to customer. If you are using IDs, you must enter the SHIP_TO_CUSTOMER_ID. If you are not using

IDs, then you must enter the SHIP_TO_CUSTOMER. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Validates whether the SHIP_TO_CUSTOMER_ID exists in RA_CUSTOMERS and RA_CUSTOMERS.STATUS is **Active**. Also, if *OE: Customer Relationships* is set to Yes, then the bill–to location must match the ordering customer or a related customer.

SHIP_ADDRESS_ID Conditionally Not Null **NUMBER**

Enter the address ID for your location (represents your complete Ship To address for the order). This value is used to derive SO_LINES_INTERFACE.SHIP_TO_SITE_USE_ID. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Must represent a valid location for the SHIP_TO_CUSTOMER and exist in RA_SITE_USES where SITE_USE_CODE is **SHIP_TO**. Validates whether the address is valid and exists in RA_ADDRESSES; must be a valid address for the SHIP_TO_CUSTOMER. Also, if *OE: Customer Relationships* is set to Yes, then the bill–to location must match the ordering customer or a related customer.

SHIP_ADDRESS1–4 **VARCHAR2(240)**

Enter the street address (4 lines) for your ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_CITY **VARCHAR2(50)**

Enter the city for your ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_STATE **VARCHAR2(50)**

Enter the state or province for your ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_COUNTY **VARCHAR2(50)**

Enter the county for your ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_COUNTRY **VARCHAR2(50)**

Enter the country for your ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_POSTAL_CODE **VARCHAR2(30)**

Enter the post code for your ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

PRICE_LIST_NAME **VARCHAR2(30)**

Enter the name of the price list for the order. If you are using IDs, you must enter the PRICE_LIST_ID. If you are not using IDs, then you must enter the PRICE_LIST_NAME, and if your PRICE_LIST_NAME is not unique, you must also enter the PRICE_LIST_ID.

Validation: SO_PRICE_LISTS.NAME

Destination: inserts SO_PRICE_LISTS.PRICE_LIST_ID into SO_HEADERS.PRICE_LIST_ID where SO_HEADER_INTERFACE.PRICE_LIST_NAME = SO_PRICE_LISTS.NAME

PRICE_LIST_ID **NUMBER**

Enter the price list ID which matches your price list. If you are using IDs, you must enter the PRICE_LIST_ID. If you are not using IDs, then you must enter the PRICE_LIST_NAME, and if your PRICE_LIST_NAME is not unique, you must also enter the PRICE_LIST_ID.

Validation: SO_PRICE_LISTS.PRICE_LIST_ID

Destination: SO_HEADERS.PRICE_LIST_ID

TERMS_NAME **VARCHAR2(15)**

Enter the name of the payment terms for the order. If you are using IDs, you must enter the TERMS_ID. If you are not using IDs, then you must enter the TERMS_NAME, and if your TERMS_NAME is not unique, you must also enter the TERMS_ID.

Validation: RA_TERMS.NAME

Destination: inserts RA_TERMS.TERM_ID into SO_HEADERS.TERM_ID where SO_HEADERS_INTERFACE.TERM_NAME = RA_TERMS.NAME

TERMS_ID **NUMBER**

Enter the payment terms ID which matches your terms name. If you are using IDs, you must enter the TERMS_ID. If you are not using IDs, then you must enter the TERMS_NAME, and if your TERMS_NAME is not unique, you must also enter the TERMS_ID.

Validation: RA_TERMS.TERM_ID

Destination: SO_HEADERS.TERM_ID

CYCLE_ID **NUMBER**

The value for this column is derived from the order type.

Validation: None

Destination: SO_HEADERS.CYCLE_ID

ENTERED_ STATE_NAME Conditionally Not Null **VARCHAR2(30)**

Enter the name of any valid status for the imported order. If you are using IDs, you should enter the ENTERED_STATE_ID. If you are not using IDs, you should enter the ENTERED_STATE_NAME, and if your ENTERED_STATE_NAME is not unique, you should also enter the ENTERED_STATE_ID. If you do not supply a value for either the ENTERED_STATE_NAME or ENTERED_STATE_ID, Manufacturing, Distribution, Sales and Service defaults the Entry Status for the order to **Entered**.

Validation: SO_RESULTS.NAME

Destination: inserts SO_RESULTS.RESULT_ID into SO_HEADERS.S1 where

SO_HEADERS_INTERFACE.ENTERED_
STATE_NAME = SO_RESULTS.NAME

ENTERED_STATE_ID Conditionally Not Null **NUMBER**

Enter the ID of any valid status for the imported order. If you are using IDs, you should enter the ENTERED_STATE_ID. If you are not using IDs, then you should enter the ENTERED_STATE_NAME, and if your ENTERED_STATE_NAME is not unique, you should also enter the ENTERED_STATE_ID. If you do not supply a value for either the ENTERED_STATE_NAME or ENTERED_STATE_ID, Manufacturing, Distribution, Sales and Service defaults the Entered State for the order to **Entered**.

Validation: SO_RESULTS.RESULT_ID

Destination: SO_HEADERS.S1

ENTERED_STATE_DATE **DATE**

Enter the date you wish to accompany the ENTERED_STATE_NAME. For example, if you enter **Booked** as the entered state name, this is the date it was booked.

Validation: Standard date validation

Destination: SO_HEADERS.S1_DATE

COMPLETE_FLAG **VARCHAR2(1)**

Enter **Y** to indicate whether this order is already closed or should not be processed, in which case the order and order lines (but no schedule details) will be imported as a complete order. Complete orders are considered for historical purposes only and can have any entered state. Enter **N** or leave this column null to indicate that this is an open order.

AGREEMENT_NAME **VARCHAR2(30)**

Enter the name of the agreement for the order. This is only required if the order type is one which requires an agreement; otherwise, it is an optional column. If you are using IDs, you can enter the AGREEMENT_ID. If you are not using IDs, then you can enter the AGREEMENT_NAME, and if your AGREEMENT_NAME is not unique, you must also enter the AGREEMENT_ID.

Validation: SO_AGREEMENTS

Destination: inserts SO_AGREEMENTS.AGREEMENT_ID into SO_HEADERS.AGREEMENT_ID where SO_HEADERS_INTERFACE.AGREEMENT_NAME = SO_AGREEMENTS.NAME

AGREEMENT_ID **NUMBER**

Enter the agreement ID that matches your AGREEMENT_NAME. If you are using IDs, you can enter the AGREEMENT_ID. If you are not using IDs, then you must enter the AGREEMENT_NAME, and if your AGREEMENT_NAME is not unique, you must also enter the AGREEMENT_ID.

Validation: SO_AGREEMENTS.AGREEMENT_ID

Destination: SO_HEADERS.AGREEMENT_ID

DATE_REQUESTED_CURRENT **DATE**

Enter the date the customer requested receipt of the order.

Validation: Must be equal to or greater than SO_HEADERS_INTERFACE.DATE_ORDERED

Destination: SO_HEADERS.DATE_REQUESTED_CURRENT

ORDERED_BY_CONTACT_FIRST_NAME **VARCHAR2(30)**

Enter the first name of the contact who placed the order. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_CONTACTS.FIRST_NAME

ORDERED_BY_CONTACT_LAST_NAME **VARCHAR2(50)**

Enter the last name of the contact who placed the order. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_CONTACTS.LAST_NAME. Both first and last names must be in RA_CONTACTS; the combination must match the ORDERED_BY_CONTACT_ID (if one is entered)

Destination: SO_HEADERS.ORDERED_BY_CONTACT_ID

ORDERED_BY_ CONTACT_ID **NUMBER**

Enter the contact ID that matches your order contact. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_CONTACTS.CONTACT_ID.

Destination: SO_HEADERS.ORDERED_BY_CONT ACT_ID

SALES_CHANNEL_CODE **VARCHAR2(30)**

Enter the name of the sales channel for the order.

Validation: SO_LOOKUPS.LOOKUP_CODE where SO_LOOKUPS.LOOKUP_TYPE = 'SALES_CHANNEL'

Destination: SO_HEADERS.SALES CHANNEL_CODE

PURCHASE_ORDER_NUM **VARCHAR2(50)**

Enter the purchase order number for the order. This is only required if the order type is one which requires a purchase order; otherwise, it is an optional column.

Validation: None

Destination: SO_HEADERS.PURCHASE_ORDER_NUM

INVOICE_TO_ CONTACT_FIRST_NAME **VARCHAR2(30)**

Enter the first name of the contact at the bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_CONTACTS.FIRST_NAME

INVOICE_TO_ CONTACT_LAST_NAME **VARCHAR2(50)**

Enter the last name of the contact at the bill–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: RA_CONTACTS.LAST_NAME. Both first and last names must be in RA_CONTACTS; the

combination must match the INVOICE_TO_CONTACT_ID (if one is entered)

INVOICE_TO_CONTACT_ID **NUMBER**

Enter the contact ID for your bill–to contact. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

The combination of contact first and last name is validated.

Validation: RA_CONTACTS.CONTACT_ID

Destination: SO_HEADERS.INVOICE_TO_CONTACT_ID

SHIP_TO_ CONTACT_FIRST_NAME **VARCHAR2(30)**

Enter the first name of the contact at the ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_TO_ CONTACT_LAST_NAME **VARCHAR2(50)**

Enter the last name of the contact at the ship–to address. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

SHIP_TO_CONTACT_ID **NUMBER**

Enter the contact ID for your ship–to contact. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

TAX_EXEMPT_NUM **VARCHAR2(30)**

If you enter E in the TAX_EXEMPT_FLAG field, you can enter an existing certificate number for the ship–to customer or enter a new, unapproved exemption certificate number.

Validation: Must be NULL if you enter Standard or Required in the Tax Exempt Flag

Destination: SO_HEADERS.TAX_EXEMPT_NUM

TAX_EXEMPT_ REASON_CODE **VARCHAR2(30)**

If you enter E in the TAX_EXEMPT_FLAG field, you must enter a valid tax exempt reason code.

Validation: AR_LOOKUPS.LOOKUP_CODE where AR_LOOKUPS.LOOKUP_TYPE = 'TAX_REASON'

Destination: SO_HEADERS.TAX_EXEMPT_REASON_CODE

TAX_EXEMPT_FLAG **VARCHAR2(1)**

Enter **E**, **S**, or **R** or accept the default. This stands for **Exempt**, **Standard** or **Required**, respectively.

Validation: If entered, must equal **E**, **S,** or **R** AR_LOOKUPS.LOOKUP_CODE where AR_LOOKUPS.LOOKUP_TYPE = 'TAX_CONTROL_FLAG'

Destination: SO_HEADERS.TAX_EXEMPT_FLAG

SHIPMENT_PRIORITY_CODE **VARCHAR2(30)**

Enter the shipment priority for the order, or use defaulting. The same defaulting used in the Sales Orders window is applied. Your standard value rule sets specify the source and priority of defaults.

Validation: SO_LOOKUPS_LOOKUP_CODE where SO_LOOKUPS.LOOKUP_TYPE = 'SHIPMENT_PRIORITY'

Destination: SO_HEADERS.SHIPMENT_PRIORITY_CODE

SHIP_METHOD_CODE **VARCHAR2(30)**

Enter the freight carrier for the order or use defaulting. The same defaulting used in the Sales Orders window is applied. Your standard value rule sets specify the source and priority of defaults.

Validation: ORG_FREIGHT.FREIGHT_CODE

Destination: SO_HEADERS.SHIP_METHOD_CODE

FREIGHT_TERMS_CODE **VARCHAR2(30)**

Enter the freight terms for the order or use defaulting. The same defaulting used in the Sales Orders window. Your standard value rule sets specify the source and priority of defaults.

Validation: SO_LOOKUPS.LOOKUP_CODE where SO_LOOKUPS.LOOKUP_TYPE = 'FREIGHT_TERMS'

Destination: SO_HEADERS.FREIGHT_TERMS_CODE

FOB_CODE **VARCHAR2(30)**

Enter the FOB for the order or use defaulting. The same defaulting used in the Sales Orders window. Your standard value rule sets specify the source and priority of defaults.

Validation: AR_LOOKUPS.LOOKUP_CODE where AR_LOOKUPS.LOOKUP_TYPE = 'FOB'

Destination: SO_HEADERS.FOB_CODE

SHIPPING_INSTRUCTIONS **VARCHAR2(240)**

Enter any shipping instructions you wish to accompany the order.

Validation: None

Destination: SO_HEADERS.SHIPPING_INSTRUCTIONS

PACKING_INSTRUCTIONS **VARCHAR2(240)**

Enter any packing instructions you wish to accompany the order.

Validation: None

Destination: SO_HEADER.PACKING_INSTRUCTIONS

DATE_SHIPPED **DATE**

Enter the date the order was shipped, if you are importing the order in the **Complete** state. If this column is populated for open orders, this information is ignored.

Validation: Must be equal to or greater than SO_HEADERS_INTERFACE.DATE_ORDERED

Destination: SO_HEADERS.DATE_SHIPPED

PAYMENT_TYPE_CODE **VARCHAR2(30)**

Enter the method of payment for the order.

Validation: AR_LOOKUPS.LOOKUP_CODE where AR_LOOKUPS.LOOKUP_TYPE = 'CASH_RECEIPT_TYPE'

Destination: SO_HEADERS.PAYMENT_CODE

PAYMENT_AMOUNT **NUMBER**

Enter the payment amount for the order.

Validation: None

Destination: SO_HEADERS.PAYMENT_AMOUNT

CHECK_NUMBER **VARCHAR2(50)**

Enter the check number for the order.

Validation: PAYMENT_TYPE_CODE must be CHECK.

Destination: SO_HEADERS.CHECK_NUMBER

CREDIT_CARD_CODE **VARCHAR2(30)**

Enter the credit card code for the type of credit card the customer is using for payment.

Validation: PAYMENT_TYPE_CODE must be CREDIT CARD and SO_LOOKUPS.LOOKUP_CODE where SO_LOOKUPS.LOOKUP_TYPE = 'CREDIT_CARD'

Destination: SO_HEADERS.CREDIT_CARD

CREDIT_CARD_ HOLDER_NAME **VARCHAR2(50)**

Enter the name of the credit card holder.

Validation: PAYMENT_TYPE_CODE must be CREDIT CARD.

Destination: SO_HEADERS.CARDHOLDER_NAME

CREDIT_CARD_NUMBER **VARCHAR2(50)**

Enter the credit card number.

Validation: PAYMENT_TYPE_CODE must be CREDIT CARD.

Destination: SO_HEADERS.CREDIT_CARD_NUM

CREDIT_CARD_EXPIRATION_DATE **DATE**

Enter the expiration date of the credit card used for payment.

Validation: PAYMENT_TYPE_CODE must be CREDIT CARD and the date must be equal to or greater than SO_HEADERS_INTERFACE.DATE_ORDERED

Destination: SO_HEADERS.CREDIT_CARD_EXPIRATION_DATE

CREDIT_CARD_APPROVAL_CODE **VARCHAR2(50)**

Enter the credit card approval number.

Validation: PAYMENT_TYPE_CODE must be CREDIT CARD.

Destination: SO_HEADERS.CREDIT_CARD_APPROVAL_CODE

ACCOUNTING_RULE **VARCHAR2(30)**

Enter the name of the accounting rule for the order, or use defaulting. If you want to enter an accounting rule, and you are using IDs, you can enter the ACCOUNTING_RULE_ID. If you are not using IDs, then you must enter the ACCOUNTING_RULE name, and if your ACCOUNTING_RULE is not unique, you must also enter the ACCOUNTING_RULE_ID.

Validation: RA_RULES.NAME where RA_RULES.TYPE = 'A'

Destination: inserts RA_RULES.RULE_ID into SO_HEADERS.ACCOUNTING_RULE where RA_RULES.NAME = SO_HEADERS.ACCOUNTING_RULE and RA_RULES.TYPE = 'A'

ACCOUNTING_RULE_ID **NUMBER**

Enter the rule ID that matches your accounting rule. If you want to enter an accounting rule, and you are using IDs, you can enter the ACCOUNTING_RULE_ID. If you are not using IDs, then you must

enter the ACCOUNTING_RULE name, and if your ACCOUNTING_RULE is not unique, you must also enter the ACCOUNTING_RULE_ID.

Validation: RA_RULES.RULE_ID where RA_RULES.TYPE = 'A'

Destination: SO_HEADERS.ACCOUNTING_RULE_ID

INVOICING_RULE **VARCHAR2(30)**

Enter the name of the invoicing rule for the order, or use defaulting. If you want to enter an invoicing rule, and you are using IDs, you can enter the INVOICING_RULE_ID. If you are not using IDs, then you must enter the INVOICING_RULE name, and if your INVOICING_RULE is not unique, you must also enter the INVOICING_RULE_ID.

Validation: RA_RULES.NAME where RA_RULES.TYPE = 'I'

Destination: inserts RA_RULES.RULE_ID into SO_HEADERS.INVOICING_RULE where RA_RULES.NAME = SO_HEADERS.INVOICING_RULE and RA_RULES.TYPE = 'I'

INVOICING_RULE_ID **NUMBER**

Enter the rule ID that matches your invoicing rule. If you want to enter an invoicing rule, and you are using IDs, you can enter the INVOICING_RULE_ID. If you are not using IDs, then you must enter the INVOICING_RULE name, and if your INVOICING_RULE is not unique, you must also enter the INVOICING_RULE_ID.

Validation: RA_RULES.RULE_ID where RA_RULES.TYPE = 'I'

Destination: SO_HEADERS.INVOICING_RULE_ID

CONTEXT **VARCHAR2(30)**

Enter the context for your descriptive flexfield, if you have enabled a context–sensitive descriptive flexfield in SO_HEADERS.

Validation: None

Destination: SO_HEADERS.CONTEXT

ATTRIBUTE1–15 **VARCHAR2(150)**

Enter any information in these 15 columns that you wish to have imported into your descriptive flexfield columns in SO_HEADERS.

Validation: None

Destination: SO_HEADERS.ATTRIBUTE1–15

INTERFACE_STATUS **VARCHAR2(1000)**

This column is used internally to store information about any invalid data.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; **Y** means an error occurred when importing this header.

SCHEDULE_STATUS_CODE **VARCHAR2(30)**

Enter the schedule status for the order; valid values include **DEMANDED**, **MATCH–RESERVE**, and **RESERVED**.

OPERATION_CODE **VARCHAR2(30)**

Enter **UPDATE** or **DELETE** for existing orders, or enter **INSERT** for new imported orders. You can also leave this column null for new imported orders.

SALESREP_NUMBER **NUMBER**

Enter the salesrep number if the salesrep name does not uniquely identify the salesperson. If you are using IDs, you must enter the SALESREP_ID. If you are not using IDs, then you must enter the SALESREP_NAME or SALESREP_NUMBER, and if your SALESREP_NAME and/or SALESREP_NUMBER is not unique, you must also enter the SALESREP_ID. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

APPLY_STANDARD_NOTES **VARCHAR2(1)**

Enter **Y** to apply standard notes automatically.

DEMAND_CLASS_CODE **VARCHAR2(30)**

Reserved for future enhancement.

ORG_ID **NUMBER**

Reserved for future enhancement.

SO_HEADER_ATTRIBUTES_INTERFACE

The following graphic summarizes the SO_HEADER_ATTRIBUTES_INTERFACE table. Database columns with a check under Conditionally Required for Booking are optional if you are importing an order with any valid status other than **Booked**. If you are importing an order with an Entry Status of **Booked**, you must enter the ID column if you are using IDs, or the name column if you are not using IDs. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 89.

SO_HEADER_ ATTRIBUTES_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
ORDER_SOURCE_ID	Number	✓			
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
OPERATION_CODE	Varchar2(30)				✓
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number				✓

Table 5 – 11 SO_HEADER_ATTRIBUTES_INTERFACE (Page 1 of 2)

SO_HEADER_ ATTRIBUTES_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
GLOBAL_ATTRIBUTE_ CATEGORY	Varchar2(30)				
GLOBAL_ATTRIBUTE1–20	Varchar2(150)				

Table 5 – 11 SO_HEADER_ATTRIBUTES_INTERFACE (Page 2 of 2)

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID. This tells OrderImport whether to use IDs or the name columns, and is used to determine which records in the interface table to process.

ORIGINAL_SYSTEM_ REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from your original feeder system. The order number of ID you enter provides you with an audit trail from OrderImport to your feeder system.

Validation: If the value you enter already exists, then the OPERATION_CODE must be **UPDATE** or **DELETE**.

Destination: SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE (This order number appears in the Sales Orders and View Orders windows.)

OPERATION_CODE **VARCHAR2(30)**

Enter **UPDATE** or **DELETE** for existing orders, or enter **INSERT** for new imported orders. You can also leave this column null for new imported orders.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; **Y** means an error occurred when importing this header.

INTERFACE_STATUS **VARCHAR2(1000)**

This column is used internally to store information about any invalid data.

CREATION_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_HEADER_ATTRIBUTES.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created the record.

Validation: None

Destination: SO_HEADER_ATTRIBUTES.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the current date on which you are entering data into your feeder system.

Validation: Standard date validation

Destination: SO_HEADER_ATTRIBUTES.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created or who most recently modified the record.

Validation: None

Destination: SO_HEADER_ATTRIBUTES.LAST_UPDATED_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Order Entry/Shipping with this data.

Validation: None

Destination: SO_HEADER_ATTRIBUTES.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER**

This column is for internal use only.

PROGRAM_ID **NUMBER**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER**

This column is for internal use only. It is populated with the concurrent manager request ID for the run of the interface each time you submit the OrderImport program. In order for records to be processed, their request ID must be null or equal to the request ID being processed.

GLOBAL_ATTRIBUTE_CATEGORY **VARCHAR2(30)**

Enter the context if you have a context–sensitive globalization flexfield.

Destination SO_HEADER_ATTRIBUTES.GLOBAL_ATTRIBUTE_CATEGORY

GLOBAL_ATTRIBUTE1–20 **VARCHAR2(150)**

Enter information that you want to pass into the globalization flexfield.

Destination SO_HEADER_ATTRIBUTES.GLOBAL_ATTRIBUTE1–20

SO_LINES_INTERFACE

The following graphic summarizes the SO_LINES_INTERFACE table. Database columns with a check under Conditionally Required for Booking are optional if you are importing an order with any valid status other than **Booked**. If you are importing an order with an Entry Status of **Booked**, you must enter the ID column if you are using IDs,

or the name column if you are not using IDs. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 95.

SO_LINES_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number				✓
PROGRAM_ APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_UPDATE_ DATE	Date				✓
REQUEST_ID	Number				✓
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
ORIGINAL_SYSTEM_ LINE_REFERENCE	Varchar2(50)	✓			
LINE_NUMBER	Number	✓			
LINE_TYPE	Varchar2(30)	✓			
UNIT_CODE	Varchar2(3)	✓			
ORDERED_QUANTITY	Number	✓			
DATE_ REQUESTED_CURRENT	Date		✓		
LIST_PRICE	Number	**C**			
SELLING_PRICE	Number	**C**			

Table 5 – 12 SO_LINES_INTERFACE (Page 1 of 4)

SO_LINES_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
INVENTORY_ITEM_ SEGMENT1 –20	Varchar2(40)	C			
INVENTORY_ITEM_ID	Number	C			
SHIPPED_QUANTITY	Number				✓
SCHEDULED_ SHIPMENT_DATE	Date				✓
LINK_TO_LINE_REF	Varchar2(50)				✓
PARENT_LINE_REF	Varchar2(50)				✓
SHIPMENT_SCHEDULE_ LINE_REF	Varchar2(50)				
SHIP_SET_NUMBER	Number				✓
SHIP_TO_SITE_USE_ID	Number			✓	
SHIP_TO_CUSTOMER	Var–char2(50)		✓		
SHIP_TO_CUSTOMER_ID	Number		✓		
SHIP_TO_CONTACT_ID	Number				✓
SHIPMENT_ PRIORITY_CODE	Varchar2(30)				✓
SHIP_METHOD_CODE	Varchar2(30)				✓
WAREHOUSE_ID	Number				✓
AGREEMENT_NAME	Varchar2(30)			✓	
AGREEMENT_ID	Number			✓	
ACCOUNTING_RULE	Varchar2(30)				✓
ACCOUNTING_RULE_ID	Number				✓
INVOICING_RULE	Varchar2(30)				✓
INVOICING_RULE_ID	Number				✓
ORDER_CATEGORY	Varchar2(30)				✓

Table 5 – 12 SO_LINES_INTERFACE (Page 2 of 4)

SO_LINES_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
ENTERED_STATE	Varchar2(30)			✓	
ENTERED_STATE_DATE	Date			✓	
COMPLETE_FLAG	Varchar2(1)			✓	
CONTEXT	Varchar2(30)				✓
ATTRIBUTE1–15	Varchar2(150)				✓
CALCULATE_PRICE	Varchar2(1)				✓
PRICING_ ATTRIBUTE1–15	Varchar2(150)				✓
PRICING_CONTEXT	Varchar2(30)				✓
PRICING_METHOD_CODE	Varchar2(30)				✓
ITEM_TYPE_CODE	Varchar2(30)				✓
OPTION_FLAG	Varchar2(1)				✓
ORDER_SOURCE_ID	Number	✓			
SHIP_ADDRESS1–4	Varchar2(240)		✓		
SHIP_ADDRESS_ID	Number		✓		
SHIP_CITY	Varchar2(50)		✓		
SHIP_COUNTRY	Varchar2(50)		✓		
SHIP_COUNTY	Varchar2(50)		✓		
SHIP_POSTAL_CODE	Varchar2(30)		✓		
SHIP_STATE	Varchar2(50)		✓		
SHIP_TO_ CONTACT_FIRST_NAME	Varchar2(30)				✓
SHIP_TO_ CONTACT_LAST_NAME	Varchar2(50)				✓
OPERATION_CODE	Varchar2(30)				✓
COMMITMENT_ID	Number				✓

Table 5 – 12 SO_LINES_INTERFACE (Page 3 of 4)

SO_LINES_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
TAX_CODE	Varchar2(50)	C			
DEMAND_CLASS_CODE	Varchar2(30)				
PROMISE_DATE	Date				
ORG_ID	Number				✓
PROJECT_ID	Number				
TASK_ID	Number				
DEMAND_STREAM_ID	Number				
INTERMEDIATE_ SHIP_TO_ID	Number				
CUSTOMER_DOCK_CODE	Varchar2(50)				
PLANNING_PROD_ SEQ_NUMBER	Varchar2(50)				
CUSTOMER_ITEM_ID	Number				
CUSTOMER_JOB	Varchar2(50)				
CUSTOMER_ PRODUCTION_LINE	Varchar2(50)				
CUSTOMER_MODEL_ SERIAL_NUMBER	Varchar2(50)				
SOURCE_TYPE_CODE	Varchar2(30)				
SOURCE_TYPE_NAME	Varchar2(80)				

Table 5 – 12 SO_LINES_INTERFACE (Page 4 of 4)

CREATION_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created the record.

Validation: None

Destination: SO_LINES.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the current date on which you are entering data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created or who most recently modified the record.

Validation: None

Destination: SO_LINES.LAST_UPDATE_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry/Shipping with this data.

Validation: None

Destination: SO_LINES.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER(15)**

This column is for internal use only.

PROGRAM_ID **NUMBER(15)**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER**

This column is for internal use only. It is populated with the concurrent manager request ID for the run of the interface each time you submit the OrderImport program. In order for records to be processed, their request ID must be null or equal to the request ID being processed.

ORIGINAL_ SYSTEM_REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from your feeder system.

Validation: Matches SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE

Destination: SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE

ORIGINAL_SYSTEM_ LINE_REFERENCE Not Null **VARCHAR2(50)**

Enter the line ID from your feeder system.

Validation: No repeated line IDs are allowed on an order.

Destination: SO_LINES.ORIGINAL_SYSTEM_LINE_ REFERENCE

LINE_NUMBER Not Null **NUMBER**

Enter the line number of the order line. The line number can be from the original system.

Validation: Must be greater than zero, an integer, and unique within the order.

Destination: SO_LINES.LINE_NUMBER

LINE_TYPE Not Null **VARCHAR2(30)**

Enter **REGULAR**. Shipment schedules are not supported.

Validation: Must be **REGULAR**

Destination: SO_LINES.LINE_TYPE_CODE

UNIT_CODE Not Null **VARCHAR2(3)**

Enter the unit of measure code for the order line.

Validation: MTL_ITEM_UOMS_VIEW.UOM_CODE

Destination: SO_LINES.UNIT_CODE

ORDERED_QUANTITY Not Null **NUMBER**

Enter the quantity ordered of this line item.

Validation: Must be greater than zero and an integer

Destination: SO_LINES.ORDERED_QUANTITY

DATE_REQUESTED_CURRENT **DATE**

Enter the date the customer requested receipt of the order or use defaulting.

Validation: Must be equal to or greater than SO_HEADERS.DATE_ORDERED

Destination: SO_LINES.DATE_REQUEST_CURRENT

LIST_PRICE (Conditionally Required) **NUMBER**

Enter the list price per unit for the order line or use automatic pricing. This column is required if you enter No in the SO_LINES_ INTERFACE.CALCULATE_PRICE column.

Validation: SO_PRICE_LIST_LINES.LIST_PRICE

Destination: SO_LINES.LIST_PRICE

SELLING_PRICE (Conditionally Required) **NUMBER**

Enter the selling price per unit for the order line or use automatic pricing. This column is required if you enter No in the SO_LINES_ INTERFACE.CALCULATE_PRICE column.

Validation: Must be less than or equal to the LIST_PRICE

Destination: SO_LINES.SELLING_PRICE

LIST_PERCENT **NUMBER**

This column is for internal use only.

SELLING_PERCENT **NUMBER**

This column is for internal use only.

INVENTORY_ITEM_SEGMENT1–20 Conditionally Required **VARCHAR2(40)**

Enter the segments of the inventory item you wish to order on the order line. As long as you enter the segments in the correct sequence, OrderImport knows which segment columns to place the values in. For example, if you use three segments, enter values in segments 1–3.

You must enter either the INVENTORY_ITEM_SEGMENT columns or INVENTORY_ITEM_ID, depending on whether you are using IDs.

Validation: Valid code combination in MTL_SYSTEM_ITEMS

Destination: insert MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID into SO_LINES.INVENTORY_ITEM_ID where SO_LINES_INTERFACE.INVENTORY_ITEM_SEGMENT1 – N = MTL_SYSTEM_ITEMS.SEGMENTS1 – N

INVENTORY_ITEM_ID Conditionally Required **NUMBER**

Enter the inventory item code combination ID which matches your combination of inventory item segments. You must enter either the INVENTORY_ITEM_SEGMENT columns or INVENTORY_ITEM_ID, depending on whether you are using IDs.

Validation: MTL_SYSTEM_ITEMS.INVENTORY_ITEM_ID

Destination: SO_LINES.INVENTORY_ITEM_ID

SHIPPED_QUANTITY **NUMBER**

Enter the shipped quantity, if you are importing this order as complete. If you enter a shipped quantity on an open order it is ignored.

Validation: None

Destination: SO_LINES.SHIPPED_QUANTITY

SCHEDULED_SHIPMENT_DATE **DATE**

Enter the scheduled shipment date for the order line or use defaulting.

Validation: Must be greater than or equal to SO_HEADERS_INTERFACE.DATE_ORDERED

Destination: SO_LINES.SCHEDULED_SHIPMENT_DATE

LINK_TO_LINE_REF **VARCHAR2(50)**

Enter the Original System Line Reference of the line for the item that is immediately above the item in this line in the bill of material (BOM) structure. This column only needs to be populated for option items, option classes, and kits. (This does not need to be entered for the top–level item.)

PARENT_LINE_REF **VARCHAR2(50)**

Enter the Original System Line Reference of the line for the item that is the top level item of the BOM to which the item in this line belongs. This only needs to be populated for option items, option classes and optional standard items. (This does not need to be entered for the top–level item.)

SHIPMENT_ SCHEDULE_LINE_REF **VARCHAR2(50)**

This column is for internal use only.

SHIP_SET_NUMBER **NUMBER**

Use this column if you want some lines to ship together (they belong to the same ship set). If two or more lines have the same ship set number, they belong to one ship set. This only needs to be populated for lines that belong to a ship set.

Validation: Must be greater than zero

Destination: SO_LINES.SHIP_SET_NUMBER

SHIP_TO_SITE_USE_ID **NUMBER**

Leave this column null; it is derived from the SHIP_ADDRESS_ID column.

Destination: SO_LINES.SHIP_TO_SITE_USE_ID

SHIP_TO_CUSTOMER **NUMBER**

Enter the name of the customer to receive the shipment, if it is different from the order. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Validates whether the SHIP_TO_CUSTOMER exists in RA_CUSTOMERS and

RA_CUSTOMERS.STATUS is **Active**. (If the customer name is not unique, you must enter an ID in the SHIP_TO_CUSTOMER_ID column.) Also, if *OE: Customer Relationships* is set to Yes, then the location must match the ordering customer or a related customer.

Destination: None

SHIP_TO_CUSTOMER_ID Conditionally Not Null **NUMBER**

Enter the customer ID of your Ship To Customer, if it is different from the order. If you are using IDs, you must enter the SHIP_TO_CUSTOMER_ID. If you are not using IDs, then you must enter the SHIP_TO_CUSTOMER. For more details on this column conditionally required for booking, see: Table 5 – 9: page 5 – 57.

Validation: Validates whether the SHIP_TO_CUSTOMER_ID exists in RA_CUSTOMERS and RA_CUSTOMERS.STATUS is **Active**. Also, if *OE: Customer Relationships* is set to Yes, then the location must match the ordering customer or a related customer.

SHIP_TO_CONTACT_ID **NUMBER**

Enter the contact ID for your ship–to contact.

Validation: RA_CONTACTS.CONTACT_ID

SHIPMENT_PRIORITY_CODE **VARCHAR2(30)**

Enter the shipment priority for the line or use defaulting. The same defaulting used in the Sales Orders window is applied. Your standard value rule sets specify the source and priority of defaults.

Validation: SO_LOOKUPS_LOOKUP_CODE where SO_LOOKUPS.LOOKUP_TYPE = 'SHIPMENT_PRIORITY'

Destination: SO_LINES.SHIPMENT_PRIORITY_CODE

SHIP_METHOD_CODE **VARCHAR2(30)**

Enter the freight carrier for the line or use defaulting. The same defaulting used in the Sales Orders window is applied. Your standard value rule sets specify the source and priority of defaults.

Validation: ORG_FREIGHT.FREIGHT_CODE

Destination: SO_LINES.SHIP_METHOD_CODE

PRICE_LIST_ID **NUMBER**

This column is for internal use only.

TERMS_ID **NUMBER**

This column is for internal use only.

WAREHOUSE_ID **NUMBER**

Enter the name of the warehouse from which you will ship this item or use defaulting. The same defaulting used in the Sales Orders window is applied. Your standard value rule sets specify the source and priority of defaults.

Validation: The warehouse specified must have the item attribute *Customer Orders Enabled* set to **Yes** for the order line item.
ORG_ORGANIZATION_DEFINITIONS.
ORGANIZATION_ID

Destination: SO_LINES.WAREHOUSE_ID

SERVICED_SEGMENT1–20 **VARCHAR2(40)**

These columns are for internal use only.

SERVICED_ITEM_ID **NUMBER**

This column is for internal use only.

SERVICED_LIST_PRICE **NUMBER**

This column is for internal use only.

SERVICE_START_DATE_ACTIVE **DATE**

This column is for internal use only.

SERVICE_END_DATE_ACTIVE **DATE**

This column is for internal use only.

CUSTOMER_PRODUCT_ID **NUMBER**

This column is for internal use only.

CP_SERVICE_ID **NUMBER**

This column is for internal use only.

TRANSACTION_TYPE_CODE **VARCHAR2(30)**

This column is for internal use only.

SERVICE_DURATION_QUANTITY **NUMBER**

This column is for internal use only.

TRANSACTION_REASON_CODE **VARCHAR2(30)**

This column is for internal use only.

TRANSACTION_COMMENTS **VARCHAR2(240)**

This column is for internal use only.

SERVICE_MASS_TXN_TEMP_ID **NUMBER**

This column is for internal use only.

AGREEMENT_NAME **VARCHAR2(30)**

Leave this column null, it is derived from SO_HEADERS_INTERFACE.AGREEMENT_NAME.

AGREEMENT_ID **NUMBER**

Leave this column null, it is derived from SO_HEADERS_INTERFACE.AGREEMENT_ID.

ACCOUNTING_RULE **VARCHAR2(30)**

Enter the name of the accounting rule only if this is a complete order. Otherwise, leave this column null and use defaulting with your standard value rule sets to ensure that the order line has an accounting rule.

Validation: RA_RULES.NAME where RA_RULES.TYPE = 'A'

Destination: inserts RA_RULES.RULE_ID into SO_LINES.ACCOUNTING_RULE_ID where RA_RULES.NAME = SO_LINES.ACCOUNTING_RULE_ID and RA_RULES.TYPE = 'A'

ACCOUNTING_RULE_ID **NUMBER**

Enter the accounting rule ID only if this is a complete order. Otherwise, leave this column null and use defaulting with your standard value rule sets to ensure that the order line has an accounting rule.

Validation: RA_RULES.RULE_ID where RA_RULES.TYPE = 'A'

Destination: SO_LINES.ACCOUNTING_RULE_ID

INVOICING_RULE **VARCHAR2(30)**

Enter the name of the invoicing rule only if this is a complete order. Otherwise, leave this column null and use defaulting with your standard value rule sets to ensure that the order line has an invoicing rule.

Validation: RA_RULES.NAME where RA_RULES.TYPE = 'I'

Destination: inserts RA_RULES.RULE_ID into SO_LINES.INVOICING_RULE_ID where RA_RULES.NAME = SO_LINES.INVOICING_RULE_ID and RA_RULES.TYPE = 'I'

INVOICING_RULE_ID **NUMBER**

Enter the invoicing rule ID only if this is a complete order. Otherwise, leave this column null and use defaulting with your standard value rule sets to ensure that the order line has an invoicing rule.

Validation: RA_RULES.RULE_ID where RA_RULES.TYPE = 'I'

Destination: SO_LINES.INVOICING_RULE_ID

SERVICED_UNIT_CODE **VARCHAR2(3)**

This column is for internal use only.

SERVICED_PRICING_ ATTRIBUTE1–15 **VARCHAR2(150)**

These columns are for internal use only.

SERVICED_PRICING_CONTEXT **VARCHAR2(30)**

This column is for internal use only.

ORDER_CATEGORY **VARCHAR2(30)**

Enter **R**, for Regular order line, in this column.

Validation: Must equal **R**.

ENTERED_STATE **VARCHAR2(30)**

Leave this column null; it is derived from SO_HEADERS_INTERFACE.

ENTERED_STATE_DATE **DATE**

Leave this column null; it is derived from SO_HEADERS_INTERFACE.

COMPLETE_FLAG **VARCHAR2(1)**

Leave this column null; it is derived from SO_HEADERS_INTERFACE.

CONTEXT **VARCHAR2(30)**

Enter the context for your descriptive flexfield attributes if you have enabled a context–sensitive descriptive flexfield in SO_LINES.

Validation: None

Destination: SO_LINES.CONTEXT

ATTRIBUTE1–15 **VARCHAR2(150)**

Enter any information you wish to pass to the descriptive flexfield columns in SO_LINES.

Validation: None

Destination: SO_LINES.ATTRIBUTE1–15

CALCULATE_PRICE **VARCHAR2(1)**

Enter **N** if you are providing a list and selling price in the LIST_PRICE and SELLING_PRICE columns, or enter **Y** if you want Manufacturing, Distribution, Sales and Service to price the order line automatically.

Attention: If you enter N and the values you enter for the list and selling prices are not equal, then you must provide a price adjustment for the difference.

PRICING_ATTRIBUTE1–15 **VARCHAR2(150)**

Enter any information you wish to pass to the pricing descriptive flexfield columns in SO_LINES.

Validation: None

Destination: SO_LINES.PRICING_ATTRIBUTE1–15

PRICING_CONTEXT **VARCHAR2(30)**

Enter the context for your pricing descriptive flexfield attributes if you have enabled a context–sensitive pricing descriptive flexfield in SO_LINES.

Validation: None

Destination: SO_LINES.PRICING_CONTEXT

PRICING_METHOD_CODE **VARCHAR2(30)**

Enter **AMOUNT** in this column.

COMPONENT_CODE **VARCHAR2(240)**

This column is for internal use only.

COMPONENT_SEQUENCE_ID **NUMBER**

This column is for internal use only.

COMPONENT_SORT_CODE **VARCHAR2(240)**

This column is for internal use only.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; **Y** means an error occurred when importing this header.

INTERFACE_STATUS **VARCHAR2(1000)**

This column is for internal use. It contains error information if OrderImport failed to import this order line.

ITEM_TYPE_CODE **VARCHAR2(30)**

Enter **CLASS**, **KIT**, **MODEL**, or **STANDARD** to indicate the type of item for the order line.

Validation: DECODE(MTL_SYSTEM_ITEMS.PICK_ COMPONENTS_FLAG, 'Y',

DECODE(MTL_SYSTEM_ITEMS.BOM_ITEM_ TYPE, 1, 'MODEL', 2, 'CLASS', 4, 'KIT'),

DECODE(MTL_SYSTEM_ITEMS.BOM_ITEM_ TYPE, 1, 'MODEL', 2, 'CLASS', 4, 'STANDARD')) = SO_LINES_INTERFACE.ITEM_TYPE_CODE

LEVEL_CODE **NUMBER**

This column is for internal use only.

OPTION_FLAG **VARCHAR2(1)**

Enter **Y** to indicate that the item on this line is an option item or option class. Enter **N** or null for base models or regular items.

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID (this should match the ORDER_SOURCE_ID of the order for this line). This column tells OrderImport whether to use IDs or the name columns and determines

which records in the interface table to process. See: Defining OrderImport Sources, *Oracle Order Entry/Shipping User's Guide*

Validation: SO_ORDER_SOURCES.ORDER_SOURCE_ID

Destination: None

SHIP_ADDRESS1–4 **VARCHAR2(240)**

Enter the street address (using up to 4 lines, one in each column) for the line–level ship–to address if it differs from the address on the header.

SHIP_ADDRESS_ID **NUMBER**

Enter the address ID for your location (represents the complete ship–to address for the order line) if it differs from the address on the header. This value is used to derive SO_LINES_INTERFACE.SHIP_TO_SITE_USE_ID.

Validation: Must represent a location contained in RA_SITE_USES where SITE_USE_CODE = 'SHIP_TO'. Validates whether the address is valid and exists in RA_ADDRESSES; must be a valid address for SO_LINES_INTERFACE.SHIP_TO_CUSTOMER, if entered; otherwise, it must be valid for SO_HEADERS_INTERFACE.SHIP_TO_CUSTOMER. Also, if *OE: Customer Relationships* is set to Yes, then the location must be a valid site use for the ship–to customer and match the ordering customer or a related customer.

SHIP_CITY **VARCHAR2(50)**

Enter the city for the line–level ship–to address if it differs from the address on the header.

SHIP_COUNTRY **VARCHAR2(50)**

Enter the country for the line–level ship–to address if it differs from the address on the header.

SHIP_COUNTY **VARCHAR2(50)**

Enter the county for the line–level ship–to address if it differs from the address on the header.

SHIP_POSTAL_CODE **VARCHAR2(30)**

Enter the post code for the line–level ship–to address if it differs from the address on the header.

SHIP_STATE **VARCHAR2(50)**

Enter the state or province for the line–level ship–to address if it differs from the address on the header.

SHIP_TO_ CONTACT_FIRST_NAME **VARCHAR2(30)**

Enter the first name of the contact at the line–level ship–to address if it differs from the contact on the header.

SHIP_TO_ CONTACT_LAST_NAME **VARCHAR2(50)**

Enter the last name of the contact at the line–level ship–to address if it differs from the contact on the header.

TERMS_NAME **VARCHAR2(15)**

This column is for internal use only.

GROUP_ID **NUMBER**

This column is for internal use only.

PERCENT_BASE_PRICE **NUMBER**

This column is for internal use only.

OPERATION_CODE **VARCHAR2(30)**

Enter **UPDATE** or **DELETE** for existing order lines, or enter **INSERT** for new imported order lines. You can also leave this column null for new imported order lines.

COMMITMENT_ID **NUMBER**

Enter the commitment ID for the order line.

Validation: RA_CUSTOMER_TRX.CUSTOMER_TRX_ID

Destination: SO_LINES.COMMITMENT_ID

TAX_CODE **VARCHAR2(50)**

If Value Added Tax is applicable, enter the tax code for the order line or accept the default. Tax code is required if the Tax Exempt Flag on the order is **R** or the Receivables Transaction Type associated with the order type requires tax calculation.

Validation: Must represent a valid tax code in Oracle Receivables
AR_VAT_TAX V, AR_SYSTEM_PARAMETERS P
WHERE V.SET_OF_BOOKS_ID =
P.SET_OF_BOOKS_ID AND ((P.TAX_METHOD =
'VAT' AND V.TAX_TYPE != 'LOCATION')
OR (P.TAX_METHOD = 'SALES_TAX')) AND
NVL(SO_LINES_INTERFACE.TAX_CODE,V.TAX_
CODE) = V.TAX_CODE AND SYSDATE
BETWEEN V.START_DATE AND
NVL(V.END_DATE,SYSDATE)

Destination: SO_LINES.TAX_CODE may be NULL if the line does not require tax.

DEMAND_CLASS_CODE **VARCHAR2(30)**

Reserved for future enhancement.

PROMISE_DATE **DATE**

Reserved for future enhancement.

ORG_ID **NUMBER**

Reserved for future enhancement.

PROJECT_ID **NUMBER**

Enter the line's project ID.

Validation MTL_PROJECTS_V.PROJECT_ID

Destination SO_LINES.PROJECT_ID

TASK_ID **NUMBER**

Enter the line's task ID.

Validation MTL_PROJECTS_V.TASK_ID

Destination SO_LINES.TASK_ID

DEMAND_STREAM_ID **NUMBER**

This column is for internal use only and is used to link order lines with Oracle Release Management data.

Validation None

Destination SO_LINES.DEMAND_STREAM_ID

INTERMEDIATE_SHIP_TO_ID **NUMBER**

This column is for internal use only and contains the ID of the line's intermediate ship–to location.

Validation RA_SITE_USES.SITE_USE_ID

Destination SO_LINES.INTERMEDIATE_SHIP_TO_ID

CUSTOMER_DOCK_CODE **VARCHAR2(50)**

This column is for internal use only and contains the dock code for lines interfaced through Oracle Release Management.

Validation None

Destination SO_LINES.CUSTOMER_DOCK_CODE

PLANNING_PROD_SEQ_NUMBER **VARCHAR2(50)**

This column is for internal use only and contains the planning production sequence number for lines interfaced through Oracle Release Management.

Validation None

Destination SO_LINES.PLANNING_PROD_SEQ_NUMBER

CUSTOMER_ITEM_ID **NUMBER**

This column is for internal use only and contains the ID of the customer item for lines interfaced through Oracle Release Management.

Validation MTL_CUSTOMER_ITEMS.CUSTOMER_ITEM_ID

Destination SO_LINES.CUSTOMER_ITEM_ID

CUSTOMER_JOB **VARCHAR2(50)**

This column is for internal use only and contains the customer's job number for lines interfaced through Oracle Release Management.

Validation None

Destination SO_LINES.CUSTOMER_JOB

CUSTOMER_ PRODUCTION_LINE **VARCHAR2(50)**

This column is for internal use only and contains the customer's production line number for lines interfaced through Oracle Release Management.

Validation None

Destination SO_LINES.CUSTOMER_PRODUCTION_LINE

CUSTOMER_MODEL_ SERIAL_NUMBER **VARCHAR2(50)**

This column is for internal use only and contains the customer's model serial number for lines interfaced through Oracle Release Management.

Validation None

Destination SO_LINES.CUSTOMER_MODEL_SERIAL_ NUMBER

SOURCE_TYPE_CODE **VARCHAR2(30)**

Enter INTERNAL or EXTERNAL to specify whether this line will be sourced internally or externally. If you leave this value null, you must either enter a value for SOURCE_TYPE_NAME or specify a source type in the Sales Orders window after you import your order. If you enter different values in SOURCE_TYPE_CODE and in SOURCE_TYPE_NAME, the value in SOURCE_TYPE_NAME is used.

Validation SO_LOOKUPS.LOOKUP_CODE where SO_LOOKUPS.LOOKUP_TYPE = 'SOURCE TYPE'

Destination SO_LINES.SOURCE_TYPE_CODE

SOURCE_TYPE_NAME **VARCHAR2(80)**

Enter INTERNAL or EXTERNAL to specify whether this line will be sourced internally or externally. If you leave this value null, you must

either enter a value for SOURCE_TYPE_CODE or specify a source type in the Sales Orders window after you import your order. If you enter different values in SOURCE_TYPE_CODE and in SOURCE_TYPE_NAME, the value in SOURCE_TYPE_NAME is used.

Validation SO_LOOKUPS.MEANING where SO_LOOKUPS.LOOKUP_TYPE = 'SOURCE TYPE'

Destination SO_LINES.SOURCE_TYPE_CODE

SO_LINE_ATTRIBUTES_INTERFACE

The following graphic summarizes the SO_LINES_INTERFACE table. Database columns with a check under Conditionally Required for Booking are optional if you are importing an order with any valid status other than **Booked**. If you are importing an order with an Entry Status of **Booked**, you must enter the ID column if you are using IDs, or the name column if you are not using IDs. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 114.

SO_LINE_ATTRIBUTES_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
ORDER_SOURCE_ID	Number	✓			
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
ORIGINAL_SYSTEM_ LINE_REFERENCE	Varchar2(50)	✓			
OPERATION_CODE	Varchar2(30)				✓
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			

Table 5 – 13 SO_LINE_ATTRIBUTES_INTERFACE (Page 1 of 2)

SO_LINE_ATTRIBUTES_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
LAST_UPDATE_LOGIN	Number				✓
PROGRAM_ APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_ UPDATE_DATE	Date				✓
REQUEST_ID	Number				✓
INDUSTRY_CONTEXT	Varchar2(30)				
INDUSTRY_ ATTRIBUTE1–15	Varchar2(150)				
GLOBAL_ATTRIBUTE_ CATEGORY	Varchar2(30)				
GLOBAL_ATTRIBUTE1–20	Varchar2(150)				

Table 5 – 13 SO_LINE_ATTRIBUTES_INTERFACE (Page 2 of 2)

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID (this should match the ORDER_SOURCE_ID of the order for this line). This column tells OrderImport whether to use IDs or the name columns and determines which records in the interface table to process. See: Defining OrderImport Sources, *Oracle Order Entry/Shipping User's Guide.*

Validation: SO_ORDER_SOURCES.ORDER_SOURCE_ID

Destination: None

ORIGINAL_ SYSTEM_REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from your feeder system.

Validation: Matches SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE

Destination: SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE

ORIGINAL_SYSTEM_LINE_REFERENCE Not Null **VARCHAR2(50)**

Enter the line ID from your feeder system.

Validation: No repeated line IDs are allowed on an order.

Destination: SO_LINES.ORIGINAL_SYSTEM_LINE_REFERENCE

OPERATION_CODE **VARCHAR2(30)**

Enter **UPDATE** or **DELETE** for existing order lines, or enter **INSERT** for new imported order lines. You can also leave this column null for new imported order lines.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; **Y** means an error occurred when importing this header.

INTERFACE_STATUS **VARCHAR2(1000)**

This column is for internal use. It contains error information if OrderImport failed to import this order line.

CREATION_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_LINE_ATTRIBUTES.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created the record.

Validation: None

Destination: SO_LINE_ATTRIBUTES.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the current date on which you are entering data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created or who most recently modified the record.

Validation: None

Destination: SO_LINES.LAST_UPDATE_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry/Shipping with this data.

Validation: None

Destination: SO_LINES.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER**

This column is for internal use only.

PROGRAM_ID **NUMBER**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER**

This column is for internal use only. It is populated with the concurrent manager request ID for the run of the interface each time you submit the OrderImport program. In order for records to be processed, their request ID must be null or equal to the request ID being processed.

INDUSTRY_CONTEXT **VARCHAR2(30)**

This column is for internal use only and contains the context of the Industry Attributes flexfield.

Validation: None

Destination: SO_LINE_ATTRIBUTES.INDUSTRY_CONTEXT

INDUSTRY_ATTRIBUTE1–15 **VARCHAR2(150)**

This column is for internal use only and contains information about the Industry Attributes flexfield segments.

Validation: None

Destination: SO_LINE_ATTRIBUTES.INDUSTRY_ATTRIBUTE1–15

GLOBAL_ATTRIBUTE_CATEGORY **VARCHAR2(30)**

Enter the context if you have a context–sensitive globalization flexfield.

Destination: SO_LINE_ATTRIBUTES.GLOBAL_ATTRIBUTE_CATEGORY

GLOBAL_ATTRIBUTE1–20 **VARCHAR2(150)**

Enter information that you want to pass into the globalization flexfield.

Destination: SO_LINE_ATTRIBUTES.GLOBAL_ATTRIBUTE1–20

SO_LINE_DETAILS_INTERFACE

The following graphic summarizes the SO_LINE_DETAILS_INTERFACE table. Database columns with a check under Conditionally Required for Booking are optional if you are importing an order with any valid status other than **Booked**. If you are importing an order with an Entry Status of **Booked**, you must enter the ID column if you are using IDs, or the name column if you are not using IDs. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 119.

SO_LINE_DETAILS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number				✓
PROGRAM_ APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_ UPDATE_DATE	Date				✓
REQUEST_ID	Number				✓
ORDER_SOURCE_ID	Number	✓			
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
ORIGINAL_SYSTEM_ LINE_REFERENCE	Varchar2(50)	✓			
QUANTITY	Number	✓			
SCHEDULE_DATE	Date		✓		
LOT_NUMBER	Varchar2(30)		✓		
SUBINVENTORY	Varchar2(10)		✓		
CUSTOMER_ REQUESTED_LOT_FLAG	Varchar2(1)				✓
CONTEXT	Varchar2(30)				✓
ATTRIBUTE1–15	Varchar2(150)				✓
INVENTORY_ITEM_ID	Number			✓	
REVISION	Varchar2(3)		✓		

Table 5 – 14 SO_LINE_DETAILS_INTERFACE (Page 1 of 2)

SO_LINE_DETAILS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
WAREHOUSE_ID	Number		✓		
DEMAND_CLASS_CODE	Varchar2(30)				

Table 5 – 14 SO_LINE_DETAILS_INTERFACE (Page 2 of 2)

CREATION_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created the record.

Validation: None

Destination: SO_LINES.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the current date on which you are entering data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created or who most recently modified the record.

Validation: None

Destination: SO_LINES.LAST_UPDATE_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry / Shipping with this data.

Validation: None

Destination: SO_LINES.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER**

This column is for internal use only.

PROGRAM_ID **NUMBER**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER**

This column is for internal use only. It is populated with the concurrent manager request ID for the run of the interface each time you submit the OrderImport program. In order for records to be processed, their request ID must be null or equal to the request ID being processed.

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID (this should match the ORDER_SOURCE_ID of the line for which this is a line detail). This column tells OrderImport whether to use IDs or the name columns and determines which records in the interface table to process.

Validation: SO_ORDER_SOURCES.ORDER_SOURCE_ID

Destination: None

ORIGINAL_ SYSTEM_REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from your feeder system.

Validation: Matches SO_HEADERS_INTERFACE.ORIGINAL_ SYSTEM_REFERENCE

Destination: SO_LINE_DETAILS.ORIGINAL_SYSTEM_REFERENCE

ORIGINAL_SYSTEM_LINE_REFERENCE Not Null **VARCHAR2(50)**

Enter the line ID from your feeder system.

Validation: Matches SO_LINES_INTERFACE.ORIGINAL_SYSTEM_LINE_REFERENCE

Destination: SO_LINE_DETAILS.ORIGINAL_SYSTEM_LINE_REFERENCE

QUANTITY Not Null **NUMBER**

Enter the quantity for the order line detail.

SCHEDULE_DATE **DATE**

Enter the schedule date for the order line detail.

LOT_NUMBER **VARCHAR2(30)**

Enter the lot number to use when releasing item from inventory if you entered **Reserved** in SO_HEADERS_INTERFACE.SCHEDULE_STATUS_CODE. Otherwise leave this column null.

SUBINVENTORY **VARCHAR2(10)**

Enter the subinventory to use when releasing item from inventory.

CUSTOMER_REQUESTED_LOT_FLAG **VARCHAR2(1)**

Enter **Y** or **N** depending on whether the customer requested a specific subinventory, lot, or revision.

CONTEXT **VARCHAR2(30)**

Enter the context for your descriptive flexfield attributes if you have enabled a context–sensitive descriptive flexfield in SO_LINE_DETAILS.

Validation: None

Destination: SO_LINE_DETAILS.CONTEXT

ATTRIBUTE1–15 **VARCHAR2(150)**

Enter any information you wish to pass to the descriptive flexfield columns in SO_LINE_DETAILS.

Validation: None

Destination: SO_LINE_DETAILS.ATTRIBUTE1 – 15

COMPONENT_SEQUENCE_ID **NUMBER**

This column is for internal use only.

INVENTORY_ITEM_ID **NUMBER**

Leave this column null; it is derived from SO_LINES_INTERFACE.

REVISION **VARCHAR2(3)**

Enter the revision number of item being scheduled if you entered **Reserved** in SO_HEADERS_INTERFACE.SCHEDULE_STATUS_CODE. Otherwise leave this column null.

SCHEDULE_LEVEL_CODE **NUMBER**

This column is for internal use only.

WAREHOUSE_ID **NUMBER**

Enter the name of the warehouse from which you will ship this item or use defaulting. The same defaulting used in the Sales Orders window is applied. Your standard value rule sets specify the source and priority of defaults.

Validation: The warehouse specified must have the item attribute *Customer Orders Enabled* set to **Yes** for the order line item.
ORG_ORGANIZATION_DEFINITIONS.
ORGANIZATION_ID

Destination: SO_LINE_DETAILS.WAREHOUSE_ID

INTERFACE_STATUS **VARCHAR2(1000)**

This column is used internally to store information about any invalid data.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; **Y** means an error occurred when importing this header.

REQUIRED_FOR_REVENUE **VARCHAR2(1)**

This column is for internal use only.

DEMAND_CLASS_CODE **VARCHAR2(30)**

Reserved for future enhancement.

SO_PRICE_ADJUSTMENTS_INTERFACE

The following graphic summarizes the SO_PRICE_ADJUSTMENTS_INTERFACE table. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 124.

SO_PRICE_ADJUSTMENTS_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number				✓
PROGRAM_ APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_ UPDATE_DATE	Date				✓

Table 5 – 15 SO_PRICE_ADJUSTMENTS_INTERFACE (Page 1 of 2)

SO_PRICE_ ADJUSTMENTS_ INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
REQUEST_ID	Number				✓
ORDER_SOURCE_ID	Number	✓			
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
ORIGINAL_SYSTEM_ LINE_REFERENCE	Varchar2(50)				✓
DISCOUNT_NAME	Varchar2(30)	C			
DISCOUNT_ID	Number	C			
PERCENT	Number	✓			
PRICING_CONTEXT	Varchar2(30)				✓
PRICING_ ATTRIBUTE1 – 15	Varchar2(150)				✓
CONTEXT	Varchar2(30)				✓
ATTRIBUTE1 – 15	Varchar2(150)				✓
DISCOUNT_LINE_ID	Number				✓
OPERATION_CODE	Varchar2(30)				✓

Table 5 – 15 SO_PRICE_ADJUSTMENTS_INTERFACE (Page 2 of 2)

CREATION_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created the record.

Validation: None

Destination: SO_LINES.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the current date on which you are entering data into your feeder system.

Validation: Standard date validation

Destination: SO_LINES.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the user who created or who most recently modified the record.

Validation: None

Destination: SO_LINES.LAST_UPDATE_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry / Shipping with this data.

Validation: None

Destination: SO_LINES.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER**

This column is for internal use only.

PROGRAM_ID **NUMBER**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER**

This column is for internal use only. It is populated with the concurrent manager request ID for the interface run each time you submit the OrderImport program. In order for records to be processed, their request IDs must be null or equal to the request ID being processed.

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID. This tells OrderImport whether to use IDs or the name columns and is used to determine which records in the interface table to process.

Validation: SO_ORDER_SOURCES

Destination: None

ORIGINAL_SYSTEM_ REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from your feeder system.

Validation: Matches SO_HEADERS_INTERFACE.ORIGINAL_ SYSTEM_REFERENCE

Destination: SO_LINES.ORIGINAL_SYSTEM_REFERENCE

ORIGINAL_SYSTEM_ LINE_REFERENCE **VARCHAR2(50)**

If the discount applies to the order line, enter the line ID from your feeder system. If this is an order–level discount, leave this column null.

Validation: Matches SO_LINES_INTERFACE.ORIGINAL_ SYSTEM_LINE_REFERENCE

Destination: SO_PRICE_ADJUSTMENTS.ORIGINAL_ SYSTEM_LINE_REFERENCE

DISCOUNT_NAME Conditionally Not Null **VARCHAR2(30)**

Enter the name of the discount that was applied to this order line if the selling price is different from the list price. If you are using IDs, you must enter values in the DISCOUNT_ID and SO_HEADERS_ INTERFACE.PRICE_LIST_ID columns. If you are not using IDs, you must enter values in the DISCOUNT_NAME and SO_HEADERS_ INTERFACE.PRICE_LIST_NAME columns.

Validation: SO_DISCOUNTS.NAME

Destination: Insert a record into SO_PRICE_ADJUSTMENTS.DISCOUNT_NAME referencing SO_DISCOUNT.DISCOUNT_NAME, where SO_LINES_INTERFACE.DISCOUNT = SO_DISCOUNTS.NAME

DISCOUNT_ID Conditionally Not Null **NUMBER**

Enter the discount ID which matches the discount name you want to use for this order line. If you enter a value here, you must also supply a value for SO_HEADERS_INTERFACE.PRICE_LIST_ID.

Validation: SO_DISCOUNTS.DISCOUNT_ID

Destination: SO_PRICE_ADJUSTMENTS.DISCOUNT_ID

Validation: SO_DISCOUNTS.DISCOUNT_LINES_ID

Destination: SO_PRICE_ADJUSTMENTS.DISCOUNT_ LINES_ID

PERCENT (Required) **NUMBER**

Enter the percent for the discount as a whole number. For example, specify ten percent as ”10”, not ”0.1”. The percent value must match the percent defined on the discount if the discount is not overridable. If you entered N in the SO_LINES_INTERFACE.CALCULATE_PRICE column, the resulting amount of this percent discount on the line must equal the difference between the list price and selling price.

Validation: SO_DISCOUNTS.PERCENT

Destination: SO_PRICE_ADJUSTMENTS.PERCENT

PRICING_CONTEXT **VARCHAR2(30)**

Enter the context for your pricing descriptive flexfield attributes if you have enabled a context–sensitive pricing descriptive flexfield in SO_PRICE_ADJUSTMENTS.

Validation: None

Destination: SO_PRICE_ADJUSTMENTS.PRICING_CONTEXT

PRICING_ATTRIBUTE1 – 15 **VARCHAR2(150)**

Enter any information you wish to pass to the pricing descriptive flexfield columns in SO_PRICE_ADJUSTMENTS.

Validation: None

Destination: SO_PRICE_ADJUSTMENTS.PRICING_ ATTRIBUTE1–15

CONTEXT **VARCHAR2(30)**

Enter the context for your descriptive flexfield attributes if you have enabled a context–sensitive descriptive flexfield in SO_PRICE_ADJUSTMENTS.

Validation: None

Destination: SO_PRICE_ADJUSTMENTS.CONTEXT

ATTRIBUTE1 – 15 **VARCHAR2(150)**

Enter any information you wish to pass to the descriptive flexfield columns in SO_PRICE_ADJUSTMENTS.

Validation: None

Destination: SO_PRICE_ADJUSTMENTS.ATTRIBUTE1–15

AUTOMATIC_FLAG **VARCHAR2(1)**

This column is for internal use only.

DISCOUNT_LINE_ID **NUMBER**

If this applies to a specific item, order type, agreement, discount, and so on, enter the discount line ID which matches the discount name you want to use for this order line.

INTERFACE_STATUS **VARCHAR2(1000)**

This column is used internally to store information about any invalid data.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; **Y** means an error occurred when importing this header.

OPERATION_CODE **VARCHAR2(30)**

Enter **UPDATE** or **DELETE** for existing price adjustments, or enter **INSERT** for new imported price adjustments. You can also leave this column null for new imported price adjustments.

SO_SALES_CREDITS_INTERFACE

The following graphic summarizes the SO_SALES_CREDITS_INTERFACE table. Database columns with a check under Conditionally Required for Booking are optional if you are importing an order with any valid status other than **Booked**. If you are importing an order with an Entry Status of **Booked**, you must enter the ID column if you are using IDs, or the name column if you are not using IDs. For more details, see: Using IDs: page 5 – 55 and Required Fields for Booked Orders: page 5 – 60.

Database columns not included in this table are for internal use only. Each column is described in detail below starting on page 5 – 124.

SO_SALES_ CREDITS_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number				✓
PROGRAM_ APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_ UPDATE_DATE	Date				✓
REQUEST_ID	Number				✓
ORIGINAL_SYSTEM_ REFERENCE	Varchar2(50)	✓			
ORIGINAL_SYSTEM_ LINE_REFERENCE	Varchar2(50)				✓
ORDER_SOURCE_ID	Number	✓			
SALESREP_NAME	Varchar2(30)		✓		
SALESREP_ID	Number		✓		

Table 5 – 16 SO_SALES_CREDITS_INTERFACE (Page 1 of 2)

SO_SALES_ CREDITS_INTERFACE Column Name	Type	Required (C indicates Condi–tionally Required)	Condi–tionally Required for Booking	Derived	Optional
SALES_CREDIT_TYPE	Varchar2(30)	C			
SALES_CREDIT_TYPE_ID	Number	C			
QUOTA_FLAG	Varchar2(1)			✓	
PERCENT	Number	✓			
CONTEXT	Varchar2(30)				✓
ATTRIBUTE1 – 15	Varchar2(150)				✓
OPERATION_CODE	Varchar2(30)				✓

Table 5 – 16 SO_SALES_CREDITS_INTERFACE (Page 2 of 2)

CREATION_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_LINE.CREATION_DATE

CREATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry / Shipping with this data.

Validation: None

Destination: SO_SALES_CREDITS.CREATED_BY

LAST_UPDATE_DATE Not Null **DATE**

Enter the date on which you originally entered the data into your feeder system.

Validation: Standard date validation

Destination: SO_SALES_CREDITS.LAST_UPDATE_DATE

LAST_UPDATED_BY Not Null **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry / Shipping with this data.

Validation: None

Destination: SO_SALES_CREDITS.LAST_UPDATE_BY

LAST_UPDATE_LOGIN **NUMBER**

Enter an identification number that you can use to identify the feeder system that is supplying Oracle Order Entry / Shipping with this data.

Validation: None

Destination: SO_SALES_CREDITS.LAST_UPDATE_LOGIN

PROGRAM_APPLICATION_ID **NUMBER(15)**

This column is for internal use only.

PROGRAM_ID **NUMBER(15)**

This column is for internal use only.

PROGRAM_UPDATE_DATE **DATE**

This column is for internal use only.

REQUEST_ID **NUMBER**

This column is for internal use only. It is populated with the concurrent manager request ID for the interface run each time you submit the OrderImport program. In order for records to be processed, their request ID must be null or equal to the request ID being processed.

ORIGINAL_SYSTEM_REFERENCE Not Null **VARCHAR2(50)**

Enter the order number or ID from the feeder system.

Validation: Must match SO_HEADERS_INTERFACE.ORIGINAL_SYSTEM_REFERENCE and SO_LINES_INTERFACE.ORIGINAL_SYSTEM_REFERENCE

Destination: None

ORIGINAL_SYSTEM_ LINE_REFERENCE **VARCHAR2(50)**

If the sales credit applies to the order line, enter the line ID from your feeder system. If this is an order–level sales credit, leave this column null.

Validation: Matches SO_LINES_INTERFACE.ORIGINAL_SYSTEM_ LINE_REFERENCE

Destination: SO_SALES_CREDITS.ORIGINAL_SYSTEM_LINE_ REFERENCE

ORDER_SOURCE_ID Not Null **NUMBER**

Enter the order import source ID. This tells OrderImport whether to use IDs or the name columns.

Validation: SO_ORDER_SOURCES

Destination: None – used to determine which records in the interface table to process and whether IDs are required.

LINE_NUMBER **NUMBER**

This column is for internal use only.

LINE_TYPE **VARCHAR2(30)**

This column is for internal use only.

SALESREP_NAME **VARCHAR2(30)**

Enter the name of the salesperson to whom you are assigning the sales credit. If you want multiple credits, you must have multiple records in the sales_credits_interface table. The total quota credit must always equal 100% for each line and for the header.

Validation: Must be Active in RA_SALESREPS.NAME for current orders; need only exist for complete orders

Destination: Inserts RA_SALESREP.SALESREP_ID into SO_SALES_CREDITS.SALESREP_ID where

SO_SALES_CREDITS_INTERFACE.SALESREP_NAME = RA_SALESREPS.NAME

SALESREP_ID **NUMBER**

Enter the salesrep ID that matches the SALESREP_NAME you entered.

Validation: RA_SALESREP.SALESREP_ID

Destination: SO_SALES_CREDITS.SALESREP_ID

SALES_CREDIT_TYPE Conditionally Not Null **VARCHAR2(30)**

Enter the name of the sales credit type you are assigning to the salesrep (such as *Quota Sales Credit* or *Nonquota Sales Credit*). If you are using IDs, you must enter the SALES_CREDIT_TYPE_ID. If you are not using IDs, you must enter the SALES_CREDIT_TYPE, and if your SALES_CREDIT_TYPE is not unique, you must also enter the SALES_CREDIT_TYPE_ID.

Validation: SO_SALES_CREDIT_TYPES.NAME

Destination: inserts SO_SALES_CREDIT_TYPES.SALES_CREDIT_TYPE_ID into SO_SALES_CREDITS where SO_SALES_CREDITS_INTERFACE.SALES_CREDIT_TYPE = SO_SALES_CREDIT_TYPES.NAME

SALES_CREDIT_TYPE_ID Conditionally Not Null **NUMBER**

Enter the SALES_CREDIT_TYPE_ID for the SALES_CREDIT_TYPE you entered. If you are using IDs, you must enter the SALES_CREDIT_TYPE_ID. If you are not using IDs, you must enter the SALES_CREDIT_TYPE, and if your SALES_CREDIT_TYPE is not unique, you must also enter the SALES_CREDIT_TYPE_ID.

Validation: SO_SALES_CREDIT_TYPES.SALES_CREDIT_TYPE_ID

Destination: SO_SALES_CREDITS.SALE_CREDIT_TYPE_ID

QUOTA_FLAG **VARCHAR2(1)**

Leave this field null; it is derived from the SALES_CREDIT_TYPE column.

PERCENT	Not Null	**NUMBER**

Enter the percent of the sales credit type you are assigning to the salesperson. The total quota credit must always equal 100% for each line and for the header.

Validation: The total must be less than or equal to 100 for quota–type sales credits. Nonquota sales credits can exceed 100.

For header–level sales credits:

```
select count(*) into :SO_SALES_CREDITS_ERRORS
    from SO_SALES_CREDITS_INTERFACE
    where NOT EXISTS
           ((select 'X' from dual
           where 100 = (select sum(SO_SALES_CREDITS.PERCENT)
           from SO_SALES_CREDITS, SO_HEADERS
           where SO_SALES_CREDITS_INTERFACE.
                  ORIGINAL_SYSTEM_REFERENCE
                  = SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE
           and SO_SALES_CREDITS_INTERFACE.
                  ORDER_SOURCE_ID
           = SO_HEADERS.ORIGINAL_SYSTEM_SOURCE_CODE
           and SO_HEADERS.HEADER_ID
           = SO_SALES_CREDITS.HEADER_ID
           and SO_HEADERS.S1 = 1   /** Booked **/
           and SO_SALES_CREDITS.LINE_ID IS NULL
           and 'Y' = (select QUOTA_FLAG
           from SO_SALES_CREDIT_TYPES
           where SO_SALES_CREDIT_TYPES.SALES_CREDIT_TYPE_ID
           = SO_SALES_CREDITS.SALES_CREDIT_TYPE_ID)))
    union ((select 'X' from dual
    where 100 >= (select nvl(sum(SO_SALES_CREDITS.PERCENT), 0)
           from SO_SALES_CREDITS, SO_HEADERS
           where SO_SALES_CREDITS_INTERFACE.
                  ORIGINAL_SYSTEM_REFERENCE
           = SO_HEADERS.ORIGINAL_SYSTEM_REFERENCE
           and SO_SALES_CREDITS_INTERFACE.ORDER_SOURCE_ID
           = SO_HEADERS.ORIGINAL_SYSTEM_SOURCE_CODE
           and SO_HEADERS.HEADER_ID
           = SO_SALES_CREDITS.HEADER_ID
           and SO_HEADERS.S1 != 1 /** Not booked **/
           and SO_SALES_CREDITS.LINE_ID IS NULL
           and 'Y' = (select QUOTA_FLAG
           from SO_SALES_CREDIT_TYPES
           where SO_SALES_CREDIT_TYPES.SALES_CREDIT_TYPE_ID
           = SO_SALES_CREDITS.SALES_CREDIT_TYPE_ID))))
```

For line–level sales credits:

The same as above except SO_SALES_CREDITS.LINE_ID is NOT NULL instead of NULL.

Destination: SO_SALES_CREDITS.PERCENT

USE_IDS_FLAG **VARCHAR2(1)**

This column is for internal use only.

CONTEXT **VARCHAR2(30)**

Enter the context for your descriptive flexfield information if you have enabled a context–sensitive descriptive flexfield in SO_SALES_CREDITS.

Validation: None

Destination: SO_SALES_CREDITS.CONTEXT

ATTRIBUTE1 – 15 **VARCHAR2(150)**

Enter any information you wish to pass to the descriptive flexfield column in SO_SALES_CREDITS.

Validation: None

Destination: SO_SALES_CREDITS.ATTRIBUTE1 – 10

INTERFACE_STATUS **VARCHAR2(1000)**

This column is used internally to store information about any invalid data.

ERROR_FLAG **VARCHAR2(1)**

This column is used internally; if a **Y** is generated it means an error occurred when importing this header.

OPERATION_CODE **VARCHAR2(30)**

Enter **UPDATE** or **DELETE** for existing sales credits, or enter **INSERT** for new imported sales credits. You can also leave this column null for new imported sales credits.

Delivery–based Ship Confirm Open Interface

The Delivery–based Ship Confirm Open Interface provides a way to load externally derived shipping data into Oracle Shipping tables and close the delivery without using the Ship Confirm–Delivery or Ship Confirm–Departure windows. The Delivery–based Ship Confirm Open Interface takes data loaded into four interface tables and:

- Validates the information contained within the interface tables
- Loads the valid data into the delivery, packed containers, picking line details, and freight charges tables
- Packs, closes, or completely backorders the delivery if requested

If you close the delivery using the Ship Confirm Open Interface and the *OE: Immediate Inventory Update* profile option is set to No, you can run the Update Shipping and Inventory Interface programs to update order lines with shipped quantities and to update inventory. If the *OE: Immediate Inventory Update* profile option is set to Yes, the Update Shipping and Inventory Interface programs run automatically when you close the delivery.

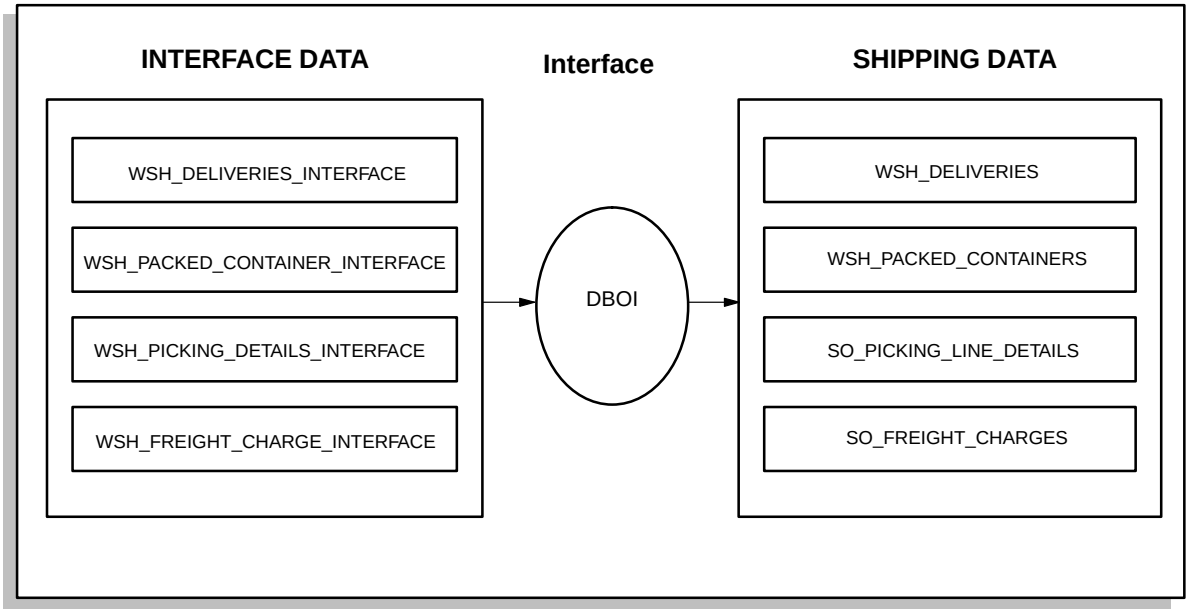
Note: These programs will only run automatically when the delivery and the departure containing the delivery are closed.

Functional Overview

The essential procedural flow in using the Delivery–based Ship Confirm Open Interface consists of:

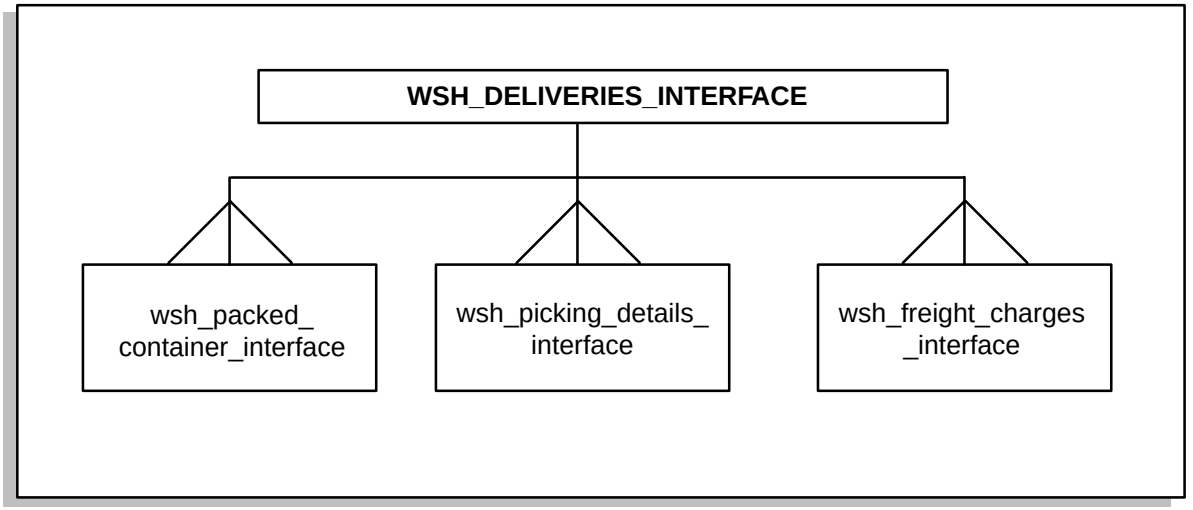
- Entering data in the delivery, packed containers, picking line details, and freight charges interface tables via an external device (such as a bar code reader).
- Running the Delivery–based Ship Confirm Open Interface concurrent process from the Standard Report Submission window.
- Checking for errors during processing. Successfully processed transactions are deleted on completion but errors are left in the interface tables to be checked, modified, or removed.

Figure 5 – 4 Transaction Flow



The Delivery–based Ship Confirm Open Interface reads data from the interface tables, performs all necessary validation on the data, loads the data into Shipping tables, and executes a pack and / or close action if requested. Ship Confirm data is divided between the four interface tables as modeled in the following entity relationship diagram:

Figure 5 – 5 WSH_DELIVERIES_INTERFACE



Transactions

The interface processes transactions, which are groupings of ship confirm data in the interface tables, for a particular delivery. Each transaction must have one record in the main delivery table (WSH_DELIVERIES_INTERFACE), but records in any of the child tables (packed containers, picking line detail, and freight charges) are optional. Types of data that can be uploaded are:

- Delivery information—either update existing attributes (departure, waybill number, picker, packer, date shipped, expected arrival date, number of boxes, weight, and unit of measure) or create a brand new delivery,
- Updated or additional packed containers,
- Updated picking line details (shipped quantities and inventory controls),
- Additional freight charges, and
- Packed and / or closed delivery with either a backorder or ship–confirm status.

For each transaction record in the WSH_DELIVERIES_INTERFACE table, the Delivery–based Ship Confirm Open Interface processes the information in the following order:

1. If packed containers exist in the WSH_PACKED_CONTAINERS_INTERFACE table for the transaction, the interface validates them, derives any data, and inserts or updates the WSH_PACKED_CONTAINERS table.
2. If picking line details exist in the WSH_PICKING_DETAILS_INTERFACE table for the transaction, the interface validates and derives any data and then updates the corresponding rows in the SO_PICKING_LINE_DETAILS table.
3. If freight charges exist in the WSH_FREIGHT_CHARGES_INTERFACE table, the interface validates and derives any data and then inserts the freight charges into the SO_FREIGHT_CHARGES table. **NOTE**: The interface will not update freight charges but always adds new ones.
4. If any of the key fields in the WSH_DELIVERIES_INTERFACE table are not null, the interface updates the corresponding fields in the WSH_DELIVERIES table for the given delivery.
5. The interface executes any action specified by the ACTION_CODE in the WSH_DELIVERIES_INTERFACE table (see below for valid values and actions).

Note: The interface will check all security rules before changing the delivery status and issue appropriate error messages if the action is not permitted.

The primary key on the WSH_DELIVERIES_INTERFACE table is the TRANSACTION_ID, which is generated using the WSH_DELIVERIES_INTEFACE_S sequence. The TRANSACTION_ID is also the foreign key to the WSH_PACKED_CONTAINERS table, the WSH_PICKING_DETAILS_INTERFACE table, and the WSH_FREIGHT_CHARGES_INTERFACE table.

The following rules apply when loading data:

- Either the DELIVERY_ID or DELIVERY_NAME must exist. If the delivery name does not exist, a new delivery will be created with this name.
- CAUTION: The interface does not validate spelling. It assumes that if the delivery name does not exist, then you are requesting it to be created. If this is not your intention, the delivery will need to be modified using Oracle Order Entry / Shipping.

- Either the CONTAINER_ID (internal Oracle sequence ID) or the SEQUENCE_NUMBER (the user input field visible in Oracle Order Entry / Shipping) is required for containers.
- Once inventory controls are updated in the shipping tables, they cannot be changed via the interface.
- Shipped quantity can only be increased, not decreased.
- Freight charges can only be inserted, not updated.

For example, say you processed a transaction that updated the picking details and defined a shipped quantity of 2. If you discovered an error and tried to correct it by resubmitting the transaction with modified data, the following would result:

- If you tried to change the inventory controls, the Delivery–based Ship Confirm Open Interface interprets the new transaction as a separate ship confirmation. As the inventory controls are different, it will insert a new record with these new inventory controls with the same shipped quantity.
- If you tried to correct the shipped quantity by defining a lower value of 1, the Delivery–based Ship Confirm Open Interface again interprets this as a separate ship confirmation. As the inventory controls are the same it will not create a new record but update the existing one by incrementing the shipped quantity by 1 (total shipped quantity now equals 3).

If you want to correct inventory control information or make changes to the quantity shipped, you must use the Ship Confirm—Deliveries window. If you want to update freight charge information, you must use the Freight Charges window.

Creating Departures

The interface is driven off deliveries and not departures. However, you may specify with which departure to associate the delivery. The interface will only create this departure if it does not already exist at the time you close the delivery. The interface will only automatically close the departure when closing the delivery when the departure is automatically created. Otherwise, you will have to close the departure from the window.

Implementing in a Non–Delivery Shipping Environment

Customers that implement shipping without deliveries must still insert a record in the WSH_DELIVERIES_INTERFACE table to maintain the transaction. We recommend that a default record is automatically inserted in here for every pick slip you confirm. The delivery name should be set to the pick slip number. This will automatically create a delivery enabling your pick slip to be processed and, at close time, the departure will automatically be created and closed. This creates a 1:1:1 relationship between departure:delivery:pickslip.

Inserting into the Delivery–based Ship Confirm Open Interface Tables

This section provides a chart for each interface table that lists all input columns, followed by a brief description of some columns requiring further explanation. Some input values have two fields, one for the internal ID (for example, Delivery_id) and another for the user–friendly term (for example, delivery_name). These are marked in the table as A and B respectively. When these are required, only one of the fields is necessary. If both fields are input, the interface will use the internal id (A) and ignore the user–friendly term (B). These tables are all available to the APPS user_id.

Delivery Interface Table

Column Name	Type	Required	Pairs	Notes
TRANSACTION_ID	Number	Yes		
PROCESS_FLAG	Number	Yes		
REQUEST_ID	Number	No		Internal use: must be null
DELIVERY_ID	Number	Yes	A	
DELIVERY_NAME	Varchar2(30)	Yes	B	
DEPARTURE_ID	Number		A	
DEPARTURE_NAME	Varchar2(30)		B	
ORGANIZATION_ID	Number	Yes	A	

Table 5 – 17 Delivery Interface Table (Page 1 of 3)

Column Name	Type	Required	Pairs	Notes
ORGANIZATION_CODE	Varchar2(3)	Yes	B	
ACTION_CODE	Number			
LOADING_ORDER_FLAG	Varchar2(2)		A	
LOADING_ORDER_FLAG_DESC	Varchar2(30)		B	
REPORT_SET_ID	Number		A	
REPORT_SET	Varchar2(30)		B	
DATE_CLOSED	Date			
SEQUENCE_NUMBER	Number			
CUSTOMER_ID	Number		A	
CUSTOMER_NUMBER	Varchar2(30)		B	
ULTIMATE_SHIP_TO_ID	Number	Yes		
INTERMEDIATE_SHIP_TO_ID	Number			
POOLED_SHIP_TO_ID	Number			
WAYBILL_NUM	Varchar2(50)			
GROSS_WEIGHT	Number			
WEIGHT_UOM_CODE	Varchar2(3)			
VOLUME	Number			
VOLUME_UOM_CODE	Varchar2(3)			
PICKED_BY_ID	Number		A	
PICKED_BY_NAME	Varchar2(30)		B	
PACKED_BY_ID	Number		A	
PACKED_BY_NAME	Varchar2(30)		B	
EXPECTED_ARRIVAL_DATE	Date			
FREIGHT_CARRIER_CODE	Varchar2(30)			
FREIGHT_TERMS_CODE	Varchar2(30)			
CURRENCY_CODE	Varchar2(15)			
FOB_CODE	Varchar2(30)			

Table 5 – 17 Delivery Interface Table (Page 2 of 3)

Column Name	Type	Required	Pairs	Notes
ERROR_EXPLANATION	Varchar2(240)			Output Only
ERROR_CODE	Varchar2(240)			Output Only
ATTRIBUTE_CATEGORY	Varchar2(150)			Flexfield Category
ATTRIBUTE1–15	Varchar2(150)			
CREATED_BY	Number	Yes		
CREATION_DATE	Date	Yes		
LAST_UPDATED_BY	Number	Yes		
LAST_UPDATE_DATE	Date	Yes		
LAST_UPDATE_LOGIN	Number	Yes		

Table 5 – 17 Delivery Interface Table (Page 3 of 3)

PROCESS_FLAG

This column indicates the processing status. The valid options are:

1. To Be Processed
2. Processed
3. Error

You must set this column to 1 to enable processing. If an error occurs while processing, this column is set to 3. You must correct the error and reset this column to 1 to process the transaction again. An error of 3 prevents all downstream processing for the delivery even if other transactions have their PROCESS_FLAG set to 1 (this is to preserve processing sequence).

ACTION_CODE

This column defines what action you wish to take on the delivery after all the transaction data has been successfully uploaded. Valid values are 1 through 5:

- NULL – No change to the delivery status (use this when you upload data but do not want to pack or close the delivery).
- 1 (Pack Entered)– Sets the delivery status to Packed. Any lines with no (null) shipped quantity automatically default to zero.

- 2 (Pack Complete) – Sets the delivery status to Packed. Any lines with no (null) shipped quantity automatically default to the requested quantity at pick time.
- 3 (Close Entered) – Same as 1 but delivery status is set to closed.
- 4 (Close Complete) – Same as 2 but delivery status is set to closed.
- 5 (Close Backorder) – Sets the delivery status to Backordered. All shipped quantities are set to zero even if there were confirmed shipped quantities.

Note: It is not necessary to set the delivery status to PACKED before closing it. The interface permits the status to proceed immediately to closed.

DELIVERY_ID / DELIVERY_NAME

These columns define the delivery . Either the DELIVERY_ID or DELIVERY_NAME should always be populated. If both are null, an error is issued.

DEPARTURE_ID / DEPARTURE_NAME

These columns define the departure and are both optional. You can use this to assign / re–assign deliveries to departures.

REPORT_SET / REPORT_SET_ID

These define the report set that you want printed when the upload completes. This must be supplied if you want a document to be printed.

WAYBILL_NUM

This column defines the waybill number. You must define a waybill number when you close a delivery containing shipped items or a waybill will be automatically assigned from the customizable WAYBILL API. This column is not required when you close a delivery if all items on the delivery are backordered.

EXPECTED_ARRIVAL_DATE

This column defines the arrival date of the delivery. This column is optional and defaults to the DATE_SHIPPED if not specified.

FREIGHT_CARRIER_CODE

Defines the Freight Carrier Code. Defaults to the freight carrier specified at order entry (SO_HEADERS_SHIP_METHOD_CODE).

UOM_CODE

Defines the unit of measure. This column is required when you close the delivery.

LOADING_ORDER_FLAG/LOADING_ORDER_FLAG_DESC

Defines the load order method.

Packed Containers Interface Table

The WSH_PACKED_CONTAINER_INTERFACE table drives the Delivery–based Ship Confirm Open Interface. The following table describes the WSH_PACKED_CONTAINER_INTERFACE table.

Column Name	Type	Required	Pairs	Notes
TRANSACTION_ID	Number	Yes		
CONTAINER_ID	Number			
SEQUENCE_NUMBER	Number			
CONTAINER_INVENTORY_ ITEM_ID	Number	Yes		
QUANTITY	Number	Yes		
PARENT_SEQUENCE_NUMBER	Number			
GROSS_WEIGHT	Number			
WEIGHT_UOM_CODE	Varchar2(30)			
MASTER_SERIAL_NUMBER	Varchar2(30)			
ORGANIZATION_ID	Number		A	
ORGANIZATION_CODE	Varchar2(3)		B	
REVISION	Varchar2(3)			

Table 5 – 18 Packed Containers Interface Table (Page 1 of 2)

Column Name	Type	Required	Pairs	Notes
LOT_NUMBER	Varchar2(30)			
SUBINVENTORY	Varchar2(10)			
INVENTORY_LOCATION_ID	Number			
SERIAL_NUMBER	Varchar2(30)			
ATTRIBUTE_CATEGORY	Varchar2(150)			Flexfield
ATTRIBUTE1–15	Varchar2(150)			
CREATED_BY	Number	Yes		
CREATION_DATE	Date	Yes		
LAST_UPDATED_BY	Number	Yes		
LAST_UPDATE_DATE	Date	Yes		
LAST_UPDATE_LOGIN	Number	Yes		
ERROR_EXPLANATION	Varchar2(240)			Output Only
ERROR_CODE	Varchar2(240)			Output Only

Table 5 – 18 Packed Containers Interface Table (Page 2 of 2)

CONTAINER_ID

Defines the Container ID. This column is a primary key used to identify existing records for updates.

SEQUENCE_NUMBER

This column is a sequence number of the container within the delivery.

PARENT_SEQUENCE_NUMBER

Defines the parent container sequence number. If the container is a master container, then this column will be null.

ORGANIZATION_ID / ORGANIZATION_CODE

Defines the Organization (warehouse) ID / Organization code from which the container is picked. This must match the organization of the delivery if supplied.

INVENTORY_LOCATION_ID

The INVENTORY_LOCATION_ID field is verified as valid in Oracle Inventory.

SUBINVENTORY/LOT_NUMBER/REVISION/ SERIAL_NUMBER/MASTER_SERIAL_NUMBER

There is currently no validation performed on these fields.

Picking Line Details Interface Table

The WSH_PICKING_DETAILS_INTERFACE is a view that physically maps onto the MTL_TRANSACTIONS_INTERFACE (MTI) table. This view represents all pick slip details information. The PROCESS_FLAG column must be set to 9. The following table describes the WSH_PICKING_DETAILS_INTERFACE view:

Column Name	Type	Required	Pairs	Notes
TRANSACTION_ID	Number	Yes		
PROCESS_FLAG	Number	Yes		
PICKING_LINE_DETAIL_ID	Number	Yes		
SOURCE_CODE	Varchar2(30)	Yes		ignored: can be anything
SOURCE_HEADER_ID	Number	Yes		ignored: can be anything
TRANSACTION_MODE	Number	Yes		ignored: can be anything
TRANSACTION_TYPE_ID	Number	Yes		ignored: can be anything
INVENTORY_ITEM_ID	Number			
WAREHOUSE_ID	Number	Yes		
SUBINVENTORY	Varchar2(30)			
LOT_NUMBER	Varchar2(30)			
REVISION	Varchar2(3)			
LOCATOR_ID	Number			

Table 5 – 19 Picking Line Details Interface Table (Page 1 of 2)

Column Name	Type	Required	Pairs	Notes
SERIAL_NUMBER	Varchar2(30)			
SHIPPED_QUANTITY	Number	Yes		
TRANSACTION_UOM	Varchar2(3)	Yes		
TRANSACTION_DATE	Date	Yes		
ATTRIBUTE_CATEGORY	Varchar2(150)			DEF for tables
ATTRIBUTE1–15	Varchar2(150)			
CREATED_BY	Number	Yes		
CREATION_DATE	Date	Yes		
LAST_UPDATED_BY	Number	Yes		
LAST_UPDATE_DATE	Date	Yes		
LAST_UPDATE_LOGIN	Number	Yes		
ERROR_EXPLANATION	Varchar2(240)			Output Only
ERROR_CODE	Varchar2(240)			Output Only
CONTAINER_SEQUENCE	Number			
CONTAINER_ID	Number			

Table 5 – 19 Picking Line Details Interface Table (Page 2 of 2)

PICKING_LINE_DETAIL_ID

This column is input from the Pick Slip Report. Use the same PICKING_LINE_DETAIL_ID for each inventory control configuration used to meet this picking line. For example, if a picking line is split between 2 subinventories, input one record for each of the subinventories. Each record must reference the original PICKING_LINE_DETAIL_ID.

PROCESS_FLAG

This column must be set to 9. Any value other than 9 may cause critical errors in the Transaction Manager.

INVENTORY_ITEM_ID

This required column must be the same as defined on the delivery.

WAREHOUSE_ID

This required column must be the same as defined on the delivery.

SUBINVENTORY

If a reservation has been placed, this column cannot be modified for the delivery. If no reservation has been placed, this column is required for validation. If null is input, the item's default subinventory is used for this column if defined.

LOT_NUMBER

If a reservation has been placed, this column cannot be modified. If no reservation has been placed, this column is only valid and required if the item is under lot control.

REVISION

If a reservation has been placed, this column cannot be modified. If no reservation has been placed, this column is only valid and required if the item is under reservation control.

LOCATOR_ID

If a reservation has been placed, this column cannot be modified. If no reservation has been placed, this column is only valid and required if the item is under location control.

SERIAL_NUMBER

This column is only valid and required if the item is under serial number control.

CONTAINER_SEQUENCE

Defines the user–friendly identifier for containers within the delivery.

Freight Charges Interface

This table loads freight charges. The following table describes the WSH_FREIGHT_CHARGES_INTERFACE table:

Column Name	Type	Required	Pairs	Notes
TRANSACTION_ID	Number	Yes		
PICKING_LINE_DETAIL_ID	Number	Yes		
CONTAINER_SEQUENCE	Number		A	
CONTAINER_ID	Number		B	
DELIVERY_FLAG	Varchar2(1)			
AMOUNT	Number	Yes		
CURRENCY_CODE	Varchar2(15)			
CURRENCY_NAME	Varchar2(15)			
FREIGHT_CHARGE_TYPE_ID	Number			
FREIGHT_CHARGE_TYPE_CODE	Varchar2(30)			
AC_ATTRIBUTE_CATEGORY	Varchar2(150)			Allowance / Charges DFF
AC_ATTRIBUTE1 – 15	Varchar2(150)			15 fields for Allowance / Charges Protected Descriptive Flexfield
ATTRIBUTE_CATEGORY	Varchar2(150)			Regular DFF
ATTRIBUTE1 –15	Varchar2(150)			Regular DFF (15 fields)
CREATED_BY	Number	Yes		
CREATION_DATE	Date	Yes		
LAST_UPDATED_BY	Number	Yes		
LAST_UPDATE_DATE	Date	Yes		
LAST_UPDATE_LOGIN	Number	Yes		

Table 5 – 20 Freight Charges Interface Table (Page 1 of 2)

Column Name	Type	Required	Pairs	Notes
ERROR_EXPLANATION	Varchar2(240)			Output Only
ERROR_CODE	Varchar2(240)			Output Only

Table 5 – 20 Freight Charges Interface Table (Page 2 of 2)

CONTAINER_ID / CONTAINER_SEQUENCE

Use these fields to identify the container, either using the internal database sequence number (container_id) or the user–friendly sequence displayed in Order Entry / Shipping (container_sequence), if you wish the charge to be associated with the container.

PICKING_LINE_DETAIL_ID

Specify the picking line detail ID if you want to associate the freight charge with the line or order.

DELIVERY_FLAG

Set this to Y if you want to associate the freight charge with the delivery. If the freight charge is not associated with a picking line, container, or delivery, then the interface will automatically associate it with the delivery regardless of this flag.

Validation

Data inserted into the interface tables must be validated and any relational integrity must be checked. The validation performed ensures that data is identical to data generated by the Ship Confirm–Delivery window. The majority of the validation aims at deriving the internal id from the user–friendly name. Other validation verifies that inventory control combinations are valid for the given item and that lines are assigned (or can be assigned) to the specified delivery. Examples of validation are given below.

- Either the DELIVERY_ID or DELIVERY_NAME must always be populated. An error is issued if both are null.
- In the WSH_PACKED_CONTAINERS_INTERFACE table, either the CONTAINER_ID or the SEQUENCE_NUMBER is required. If

CONTAINER_ID is null and SEQUENCE_NUMBER does not exist in WSH_PACKED_CONTAINERS, then a new container is created.

- If the delivery line in the WSH_PICKING_DETAILS_INTERFACE table has a container associated with it, then the container, identified by CONTAINER_SEQUENCE or CONTAINER_ID, must exist in WSH_PACKED_CONTAINERS (which may have been added during the processing of the container interface table).
- The value of PICKED_BY_NAME / PICKED_BY_ID must be valid in FND_USER.
- The value of PACKED_BY_NAME / PACKED_BY_ID must be valid in FND_USER.
- REPORT_SET / REPORT_SET_ID must be valid in SO_REPORT_SETS.
- UNIT_OF_MEASURE / UOM_CODE must be valid in MTL_UNITS_OF_MEASURE.
- LOAD_ORDER_FLAG / LOAD_ORDER_FLAG_DESC must be valid in SO_LOOKUPS for type LOADING_ORDER.
- WAREHOUSE_ID must be valid in ORG_ORGANIZATION_DEFINITIONS.
- FREIGHT_CARRIER_CODE must be valid in ORG_FREIGHT for the organization.
- FREIGHT_CHARGE_TYPE_ID / FREIGHT_CHARGE_TYPE_DESC must be valid in SO_FREIGHT_CHARGE_TYPES as FREIGHT_CHARGE_TYPE_ID / NAME.
- CURRENCY_CODE / CURRENCY_NAME must be either freight currency or order currency.
- EXPECTED_ARRIVAL_DATE must be greater than or equal to the DATE_SHIPPED if specified.
- Subinventory must be valid for the item and not disabled. Lot must be valid for the item, organization, and subinventory. Serial number must be valid for the item, organization, revision, or lot when using predefined serial number control. Serial numbers must be unique. Locator must be valid for the item and not disabled.

- The total shipped quantity cannot exceed the requested quantity. Quantities cannot be negative and, when referring to serial numbers, the quantity must be 1.

Viewing Failed Transactions

Transactions that fail the validation process are labeled with a PROCESS_FLAG = 3 on the delivery record (WSH_DELIVERIES_INTERFACE). The ERROR_CODE and ERROR_EXPLANATION columns will be populated with the name and description of the error on the record which caused the error. If the error did not occur on the delivery record, the ERROR_CODE column will indicate which of the details tables caused the error and the ERROR_CODE/ERROR_EXPLANATION within those tables will explain the error.

The concurrent program will complete with a warning status should errors occur. The log file will list the transactions with the error explanations. A completion status of ERROR is reserved for system errors (errors that prohibit any complete execution of the code).

Fixing Failed Transactions

If you encounter failed transactions, you can delete the failed records, correct the errors in the external feeder system, and resubmit them. Alternatively, you can update the interface records in the interface table using SQL*PLUS. When you resubmit updated transactions for processing, all validation will be performed again. When you resubmit the transaction, make sure that the PROCESS_FLAG column is set to 1.

Recovering from a Failed Concurrent Program

Should the concurrent program fail because of a system error, the completion status will be ERROR. Examine your log file for any messages. When you resubmit the request, make sure the REQUEST_ID field is null on all records you want to process (the field is used internally while processing—premature termination may prevent the interface from resetting it to null).

Shipping Transaction Manager

Your Shipping Transaction Manager must be up and running if you wish to process Update Shipping and Inventory Interface online. The steps involved for starting the Shipping Transaction Manager using the 10SC version of Order Entry / Shipping are listed below.

Attention: The Internal Manager must be started for the Shipping Transaction Manager to run.

Order Entry/Shipping 10SC Version

To start the Shipping Transaction Manager:

1. Change to the System Administrator responsibility.
2. Navigate to the Administer Concurrent Managers window (select Concurrent:Manager:Administer).
3. Select the Shipping Transaction manager option.
4. Select the Activate button.
5. The Activate button is only available when the Target field is set to zero. The Target field determines the maximum number of manager processes that can be active for the Shipping Transaction Manager. You can define the Target field (the maximum number of processes) in the Concurrent Manager window. If the Target field is already defined (has a quantity greater than zero defined), the Shipping Transaction Manager will run automatically when the Internal Manager is started.

To define the maximum number of manager processes:

1. Change to the System Administrator responsibility.
2. Navigate to the Concurrent Manager window (select Concurrent:Manager:Define).
3. Query the Shipping Transaction Manager.
4. Select the Work Shifts button.
5. Navigate to the Work Shifts window.
6. Define the appropriate number of Processes.
7. Save your work.

Attention: If you want to run the Update Shipping Information and Inventory Interface concurrent programs online, you must start the Shipping Transaction Manager.

CHAPTER

6

Oracle Purchasing or Oracle Public Sector Purchasing Open Interfaces

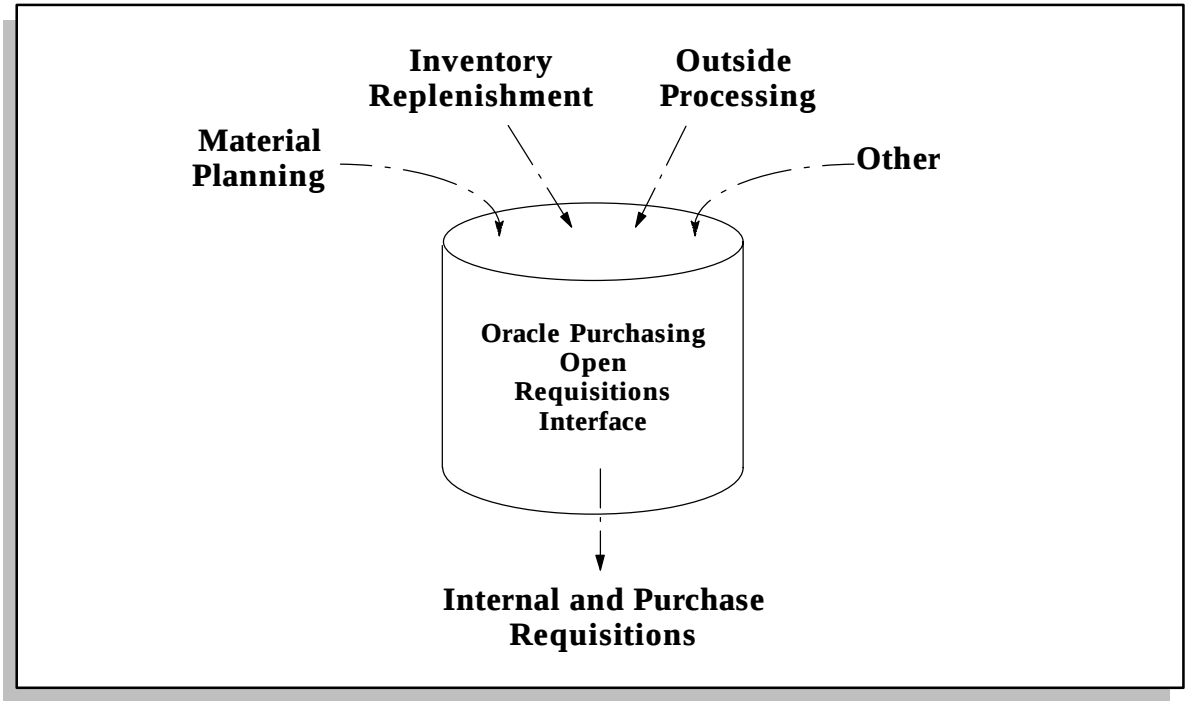
This chapter contains information about the following Oracle Purchasing or Oracle Public Sector Purchasing open interfaces:

Open Requisitions Interface

You can automatically import requisitions from other Oracle Applications or your existing non–Oracle systems using the Open Requisitions 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 Interface so that you can integrate other applications with Purchasing.

Figure 6 – 1



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 and/or the PO_REQ_DIST_INTERFACE 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 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 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 or the PO_REQ_DIST_INTERFACE table. In the Requisition Import program, you specify a parameter of Yes or No for Multiple Distributions. This information is entered into a column in the PO_REQUISITIONS_INTERFACE table as 'Y' for Yes, 'N' for No, or Null. If it's Y, then Requisition Import looks for distribution information in the PO_REQ_DISTRIBUTIONS table. If it's N, then Requisition Import looks for distribution information in the PO_REQUISITIONS_INTERFACE table. Null is treated as No (N). If you are importing multiple distributions for at least one requisition line, you should load the distribution information into the PO_REQ_DIST_INTERFACE table, enter a value of Yes in the Requisition Import program, and set the column MULTI_DISTRIBUTIONS in the PO_REQUISITIONS_INTERFACE table to Y.

Note: If you enter an Import Source parameter of Oracle Master Scheduling/MRP, Oracle Order Entry, or Oracle Inventory (INV), you must enter No for Multiple Distributions. If you enter an Import Source of Oracle Web Requisitions (ICX), you must enter Yes for Multiple Distributions.

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

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 table is Y, Requisition Import also checks for any records in the PO_REQUISITIONS_INTERFACE table without corresponding distribution information in the PO_REQ_DIST_INTERFACE 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 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 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 or the PO_REQ_DIST_INTERFACE 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 both the requisition and requisition distributions 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 Open Requisitions Interface. If you are using requisition encumbrance, approved requisitions that you import automatically become pre–approved. The requisition approval process in Purchasing then attempts to automatically reserve funds for the document at the beginning of the approval process. If it cannot, then Purchasing reserves the funds when an approver with enough authority to reserve the funds approves the document.

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 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 Customers or Customer Summary windows.

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*

Associating Ship–to and Receiving Locations, *Oracle Order Entry/Shipping 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.

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
TRANSACTION_ID	Number		✓	
PROCESS_FLAG	Varchar2		✓	
REQUEST_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
LAST_UPDATED_BY	Number		✓	
LAST_UPDATE_DATE	Date		✓	
LAST_UPDATE_LOGIN	Number		✓	
CREATION_DATE	Date			✓
CREATED_BY	Number			✓
INTERFACE_SOURCE_CODE	Varchar2	✓		
INTERFACE_SOURCE_LINE_ID	Number			✓
BATCH_ID	Number			✓
GROUP_CODE	Varchar2			✓
SOURCE_TYPE_CODE	Varchar2	*conditionally*	*conditionally*	

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 1 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
REQUISITION_TYPE	Varchar2		✓	
DESTINATION_TYPE_CODE	Varchar2	✓		
AUTHORIZATION_STATUS	Varchar2	✓		
PREPARER_ID	Number	✓	*conditionally*	
PREPARER_NAME	Varchar2			✓
APPROVER_ID	Number		✓	
APPROVER_NAME	Varchar2			✓
APPROVAL_PATH_ID	Number			✓
REQUISITION_HEADER_ID	Number		✓	
REQUISITION_LINE_ID	Number		✓	
REQ_DISTRIBUTION_ID	Number		✓	
REQ_NUMBER_SEGMENT1	Varchar2			✓
REQ_NUMBER_SEGMENT2	Varchar2			✓
REQ_NUMBER_SEGMENT3	Varchar2			✓
REQ_NUMBER_SEGMENT4	Varchar2			✓
REQ_NUMBER_SEGMENT5	Varchar2			✓
HEADER_DESCRIPTION	Varchar2			✓
HEADER_ATTRIBUTE_CATEGORY	Varchar2			✓
HEADER_ATTRIBUTE1	Varchar2			✓
HEADER_ATTRIBUTE2	Varchar2			✓
HEADER_ATTRIBUTE3	Varchar2			✓
HEADER_ATTRIBUTE4	Varchar2			✓
HEADER_ATTRIBUTE5	Varchar2			✓
HEADER_ATTRIBUTE6	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 2 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
HEADER_ATTRIBUTE7	Varchar2			✓
HEADER_ATTRIBUTE8	Varchar2			✓
HEADER_ATTRIBUTE9	Varchar2			✓
HEADER_ATTRIBUTE10	Varchar2			✓
HEADER_ATTRIBUTE11	Varchar2			✓
HEADER_ATTRIBUTE12	Varchar2			✓
HEADER_ATTRIBUTE13	Varchar2			✓
HEADER_ATTRIBUTE14	Varchar2			✓
HEADER_ATTRIBUTE15	Varchar2			✓
URGENT_FLAG	Varchar2			✓
RFQ_REQUIRED_FLAG	Varchar2			✓
JUSTIFICATION	Varchar2			✓
NOTE_TO_BUYER	Varchar2			✓
NOTE_TO_RECEIVER	Varchar2			✓
NOTE_TO_APPROVER	Varchar2			✓
ITEM_ID	Number	*conditionally*	*conditionally*	
ITEM_SEGMENT1	Varchar2			✓
ITEM_SEGMENT2	Varchar2			✓
ITEM_SEGMENT3	Varchar2			✓
ITEM_SEGMENT4	Varchar2			✓
ITEM_SEGMENT5	Varchar2			✓
ITEM_SEGMENT6	Varchar2			✓
ITEM_SEGMENT7	Varchar2			✓
ITEM_SEGMENT8	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 3 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
ITEM_SEGMENT9	Varchar2			✓
ITEM_SEGMENT10	Varchar2			✓
ITEM_SEGMENT11	Varchar2			✓
ITEM_SEGMENT12	Varchar2			✓
ITEM_SEGMENT13	Varchar2			✓
ITEM_SEGMENT14	Varchar2			✓
ITEM_SEGMENT15	Varchar2			✓
ITEM_SEGMENT16	Varchar2			✓
ITEM_SEGMENT17	Varchar2			✓
ITEM_SEGMENT18	Varchar2			✓
ITEM_SEGMENT19	Varchar2			✓
ITEM_SEGMENT20	Varchar2			✓
ITEM_DESCRIPTION	Varchar2		✓	
ITEM_REVISION	Varchar2			✓
CATEGORY_ID	Number	*conditionally*	*conditionally*	
CATEGORY_SEGMENT1	Varchar2			✓
CATEGORY_SEGMENT2	Varchar2			✓
CATEGORY_SEGMENT3	Varchar2			✓
CATEGORY_SEGMENT4	Varchar2			✓
CATEGORY_SEGMENT5	Varchar2			✓
CATEGORY_SEGMENT6	Varchar2			✓
CATEGORY_SEGMENT7	Varchar2			✓
CATEGORY_SEGMENT8	Varchar2			✓
CATEGORY_SEGMENT9	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 4 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
CATEGORY_SEGMENT10	Varchar2			✓
CATEGORY_SEGMENT11	Varchar2			✓
CATEGORY_SEGMENT12	Varchar2			✓
CATEGORY_SEGMENT13	Varchar2			✓
CATEGORY_SEGMENT14	Varchar2			✓
CATEGORY_SEGMENT15	Varchar2			✓
CATEGORY_SEGMENT16	Varchar2			✓
CATEGORY_SEGMENT17	Varchar2			✓
CATEGORY_SEGMENT18	Varchar2			✓
CATEGORY_SEGMENT19	Varchar2			✓
CATEGORY_SEGMENT20	Varchar2			✓
QUANTITY	Number	✓		
UNIT_PRICE	Number		✓	
CHARGE_ACCOUNT_ID	Number	✓	*conditionally*	
CHARGE_ACCOUNT_SEGMENT1	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT2	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT3	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT4	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT5	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT6	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT7	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT8	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT9	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT10	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 5 of 11)

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

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 6 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
UOM_CODE	Varchar2			✓
LINE_TYPE_ID	Number		✓	
LINE_TYPE	Varchar2			✓
UN_NUMBER_ID	Number		*conditionally*	
UN_NUMBER	Varchar2			✓
HAZARD_CLASS_ID	Number		*conditionally*	
HAZARD_CLASS	Varchar2			✓
MUST_USE_SUGG_VENDOR_FLAG	Varchar2			✓
REFERENCE_NUM	Varchar2			✓
SOURCE_ORGANIZATION_ID	Number		*conditionally*	
SOURCE_ORGANIZATION_CODE	Varchar2			✓
SOURCE_SUBINVENTORY	Varchar2			✓
DESTINATION_ORGANIZATION_ID	Number	✓	*conditionally*	
DESTINATION_ORGANIZATION_CODE	Varchar2			✓
DESTINATION_SUBINVENTORY	Varchar2	*conditionally*		
DELIVER_TO_LOCATION_ID	Number	✓	*conditionally*	
DELIVER_TO_LOCATION_CODE	Varchar2			✓
DELIVER_TO_REQUESTOR_ID	Number	✓	*conditionally*	
DELIVER_TO_REQUESTOR_NAME	Varchar2			✓
AUTOSOURCE_FLAG	Varchar2			✓
AUTOSOURCE_DOC_HEADER_ID	Number		*conditionally*	
AUTOSOURCE_DOC_LINE_NUM	Number		*conditionally*	
DOCUMENT_TYPE_CODE	Varchar2		*conditionally*	
SUGGESTED_BUYER_ID	Number		*conditionally*	

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 7 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
SUGGESTED_BUYER_NAME	Varchar2			✓
SUGGESTED_VENDOR_ID	Number		*conditionally*	
SUGGESTED_VENDOR_NAME	Varchar2			✓
SUGGESTED_VENDOR_SITE_ID	Number		*conditionally*	
SUGGESTED_VENDOR_SITE	Varchar2			✓
SUGGESTED_VENDOR_CONTACT_ID	Number		*conditionally*	
SUGGESTED_VENDOR_CONTACT	Varchar2		*conditionally*	
SUGGESTED_VENDOR_PHONE	Varchar2		*conditionally*	
SUGGESTED_VENDOR_ITEM_NUM	Varchar2			✓
LINE_ATTRIBUTE_CATEGORY	Varchar2			✓
LINE_ATTRIBUTE1	Varchar2			✓
LINE_ATTRIBUTE2	Varchar2			✓
LINE_ATTRIBUTE3	Varchar2			✓
LINE_ATTRIBUTE4	Varchar2			✓
LINE_ATTRIBUTE5	Varchar2			✓
LINE_ATTRIBUTE6	Varchar2			✓
LINE_ATTRIBUTE7	Varchar2			✓
LINE_ATTRIBUTE8	Varchar2			✓
LINE_ATTRIBUTE9	Varchar2			✓
LINE_ATTRIBUTE10	Varchar2			✓
LINE_ATTRIBUTE11	Varchar2			✓
LINE_ATTRIBUTE12	Varchar2			✓
LINE_ATTRIBUTE13	Varchar2			✓
LINE_ATTRIBUTE14	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 8 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
LINE_ATTRIBUTE15	Varchar2			✓
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*	
NOTE6_ID	Number		*conditionally*	
NOTE7_ID	Number		*conditionally*	
NOTE8_ID	Number		*conditionally*	
NOTE9_ID	Number		*conditionally*	
NOTE10_ID	Number		*conditionally*	
NOTE1_TITLE	Varchar2			✓
NOTE2_TITLE	Varchar2			✓
NOTE3_TITLE	Varchar2			✓
NOTE4_TITLE	Varchar2			✓
NOTE5_TITLE	Varchar2			✓
NOTE6_TITLE	Varchar2			✓
NOTE7_TITLE	Varchar2			✓
NOTE8_TITLE	Varchar2			✓
NOTE9_TITLE	Varchar2			✓
NOTE10_TITLE	Varchar2			✓
DIST_ATTRIBUTE_CATEGORY	Varchar2			✓
DISTRIBUTION_ATTRIBUTE1	Varchar2			✓
DISTRIBUTION_ATTRIBUTE2	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 9 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
DISTRIBUTION_ATTRIBUTE3	Varchar2			✓
DISTRIBUTION_ATTRIBUTE4	Varchar2			✓
DISTRIBUTION_ATTRIBUTE5	Varchar2			✓
DISTRIBUTION_ATTRIBUTE6	Varchar2			✓
DISTRIBUTION_ATTRIBUTE7	Varchar2			✓
DISTRIBUTION_ATTRIBUTE8	Varchar2			✓
DISTRIBUTION_ATTRIBUTE9	Varchar2			✓
DISTRIBUTION_ATTRIBUTE10	Varchar2			✓
DISTRIBUTION_ATTRIBUTE11	Varchar2			✓
DISTRIBUTION_ATTRIBUTE12	Varchar2			✓
DISTRIBUTION_ATTRIBUTE13	Varchar2			✓
DISTRIBUTION_ATTRIBUTE14	Varchar2			✓
DISTRIBUTION_ATTRIBUTE15	Varchar2			✓
GOVERNMENT_CONTEXT	Varchar2			✓
GL_DATE	Date		*conditionally*	
USSGL_TRANSACTION_CODE	Varchar2			✓
PREVENT_ENCUMBRANCE_FLAG	Varchar2		✓	
CURRENCY_CODE	Varchar2			✓
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			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 10 of 11)

PO_REQUISITIONS_INTERFACE Column Name	Type	Required	Derived	Optional
WIP_OPERATION_SEQ_NUM	Number			✓
WIP_RESOURCE_SEQ_NUM	Number			✓
WIP_REPETITIVE_SCHEDULE_ID	Number	*conditionally*		
BOM_RESOURCE_ID	Number	*conditionally*		
EXPENDITURE_ORGANIZATION_ID	Number	*conditionally*		
EXPENDITURE_TYPE	Varchar2	*conditionally*		
PROJECT_ACCOUNTING_CONTEXT	Varchar2			✓
PROJECT_ID	Number	*conditionally*	*conditionally*	
PROJECT_NUM	Varchar2			✓
TASK_ID	Number	*conditionally*	*conditionally*	
TASK_NUM	Varchar2			✓
EXPENDITURE_ITEM_DATE	Date			✓
TRANSACTION_REASON_CODE	Varchar2			✓
ORG_ID	Number	*conditionally*		
ALLOCATION_TYPE	Varchar2			✓
ALLOCATION_VALUE	Number			✓
MULTI_DISTRIBUTIONS	Varchar2			✓
REQ_DIST_SEQUENCE_ID	Number			✓
KANBAN_CARD_ID	Number			✓
EMERGENCY_PO_NUM	Varchar2			✓

Table 6 – 1 Open Requisitions Interface (Requisitions) (Page 11 of 11)

Requisition Distributions Interface Table Description

The following graphic describes the requisition distributions interface table.

PO_REQ_DIST_INTERFACE Column Name	Type	Required	Derived	Optional
PROJECT_ACCOUNTING_CONTEXT	Varchar2			✓
EXPENDITURE_ORGANIZATION_ID	Number	*conditionally*		
PROJECT_ID	Number	*conditionally*	*conditionally*	
TASK_ID	Number	*conditionally*	*conditionally*	
EXPENDITURE_ITEM_DATE	Date			✓
GL_DATE	Date		*conditionally*	
DIST_ATTRIBUTE_CATEGORY	Varchar2			✓
DISTRIBUTION_ATTRIBUTE1	Varchar2			✓
DISTRIBUTION_ATTRIBUTE2	Varchar2			✓
DISTRIBUTION_ATTRIBUTE3	Varchar2			✓
DISTRIBUTION_ATTRIBUTE4	Varchar2			✓
DISTRIBUTION_ATTRIBUTE5	Varchar2			✓
DISTRIBUTION_ATTRIBUTE6	Varchar2			✓
DISTRIBUTION_ATTRIBUTE7	Varchar2			✓
DISTRIBUTION_ATTRIBUTE8	Varchar2			✓
DISTRIBUTION_ATTRIBUTE9	Varchar2			✓
DISTRIBUTION_ATTRIBUTE10	Varchar2			✓
DISTRIBUTION_ATTRIBUTE11	Varchar2			✓
DISTRIBUTION_ATTRIBUTE12	Varchar2			✓
DISTRIBUTION_ATTRIBUTE13	Varchar2			✓
DISTRIBUTION_ATTRIBUTE14	Varchar2			✓

Table 6 – 2 Open Requisitions Interface (Distributions) (Page 1 of 5)

PO_REQ_DIST_INTERFACE Column Name	Type	Required	Derived	Optional
DISTRIBUTION_ATTRIBUTE15	Varchar2			✓
CHARGE_ACCOUNT_ID	Number	✓	*conditionally*	
CHARGE_ACCOUNT_SEGMENT1	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT2	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT3	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT4	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT5	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT6	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT7	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT8	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT9	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT10	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT11	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT12	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT13	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT14	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT15	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT16	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT17	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT18	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT19	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT20	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT21	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT22	Varchar2			✓

Table 6 – 2 Open Requisitions Interface (Distributions) (Page 2 of 5)

PO_REQ_DIST_INTERFACE Column Name	Type	Required	Derived	Optional
CHARGE_ACCOUNT_SEGMENT23	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT24	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT25	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT26	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT27	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT28	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT29	Varchar2			✓
CHARGE_ACCOUNT_SEGMENT30	Varchar2			✓
GROUP_CODE	Varchar2			✓
PROJECT_NUM	Varchar2			✓
TASK_NUM	Varchar2			✓
EXPENDITURE_TYPE	Varchar2	*conditionally*		
DIST_SEQUENCE_ID	Number	*conditionally*		
ALLOCATION_TYPE	Varchar2	*conditionally*		
ALLOCATION_VALUE	Number	*conditionally*		
BATCH_ID	Number			✓
DISTRIBUTION_NUMBER	Number	✓		
ITEM_ID	Number	*conditionally*	*conditionally*	
ACCRUAL_ACCOUNT_ID	Number		✓	
VARIANCE_ACCOUNT_ID	Number		✓	
BUDGET_ACCOUNT_ID	Number		✓	
USSGL_TRANSACTION_CODE	Varchar2			✓
GOVERNMENT_CONTEXT	Varchar2			✓
PREVENT_ENCUMBRANCE_FLAG	Varchar2			✓

Table 6 – 2 Open Requisitions Interface (Distributions) (Page 3 of 5)

PO_REQ_DIST_INTERFACE Column Name	Type	Required	Derived	Optional
TRANSACTION_ID	Number		✓	
PROCESS_FLAG	Varchar2		✓	
REQUEST_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
LAST_UPDATED_BY	Number		✓	
LAST_UPDATE_DATE	Date		✓	
LAST_UPDATE_LOGIN	Number		✓	
CREATION_DATE	Date			✓
DESTINATION_ORGANIZATION_ID	Number	✓	*conditionally*	
DESTINATION_SUBINVENTORY	Varchar2	*conditionally*		
DESTINATION_TYPE_CODE	Varchar2	✓		
CREATED_BY	Number			✓
INTERFACE_SOURCE_CODE	Varchar2	✓		✓
INTERFACE_SOURCE_LINE_ID	Number			✓
REQUISITION_HEADER_ID	Number		✓	
REQUISITION_LINE_ID	Number		✓	
REQ_DISTRIBUTION_ID	Number		✓	
REQ_NUMBER_SEGMENT1	Varchar2			✓
REQ_NUMBER_SEGMENT2	Varchar2			✓
REQ_NUMBER_SEGMENT3	Varchar2			✓
REQ_NUMBER_SEGMENT4	Varchar2			✓
REQ_NUMBER_SEGMENT5	Varchar2			✓

Table 6 – 2 Open Requisitions Interface (Distributions) (Page 4 of 5)

PO_REQ_DIST_INTERFACE Column Name	Type	Required	Derived	Optional
QUANTITY	Number	*conditionally*	*conditionally*	
ORG_ID	Number	*conditionally*		

Table 6 – 2 Open Requisitions Interface (Distributions) (Page 5 of 5)

Required Data

You must always enter values for the following required columns when you load rows into the PO_REQUISITIONS_INTERFACE 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 when you load rows into the PO_REQ_DIST_INTERFACE 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 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: page 6 – 27.

In the PO_REQ_DIST_INTERFACE 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 and PO_REQ_DIST_INTERFACE 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 Organization Parameters window in Oracle Inventory, 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

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

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

For additional information on conditionally required columns, see: Validation: page 6 – 27.

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. (Vendor 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 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 vendor–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 vendor–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 vendor–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, vendor sourcing fails and the UNIT_PRICE is defaulted from the item master for vendor requisition lines and from the CST_ITEM_COSTS_FOR_GL_VIEW for internal requisitions.

In the PO_REQ_DIST_INTERFACE 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 and PO_REQ_DIST_INTERFACE 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 Open Requisitions 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 information, 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 11* 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.

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 table and the requisition distributions interface table 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:

Column Name	Null?	Type
LINE_ID	NOT NULL	NUMBER(15)
QUANTITY		NUMBER

Column Name	Null?	Type
NEED_BY_DATE		DATE
PROCESS_ID		NUMBER(15)
LAST_UPDATE_DATE		DATE
LAST_UPDATED_BY		NUMBER(15)
LAST_UPDATE_LOGIN		NUMBER(15)
CREATION_DATE		DATE
CREATED_BY		NUMBER(15)

Table 6 – 3 (Page 2 of 2)

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

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

Table 6 – 4 (Page 1 of 1)

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 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.

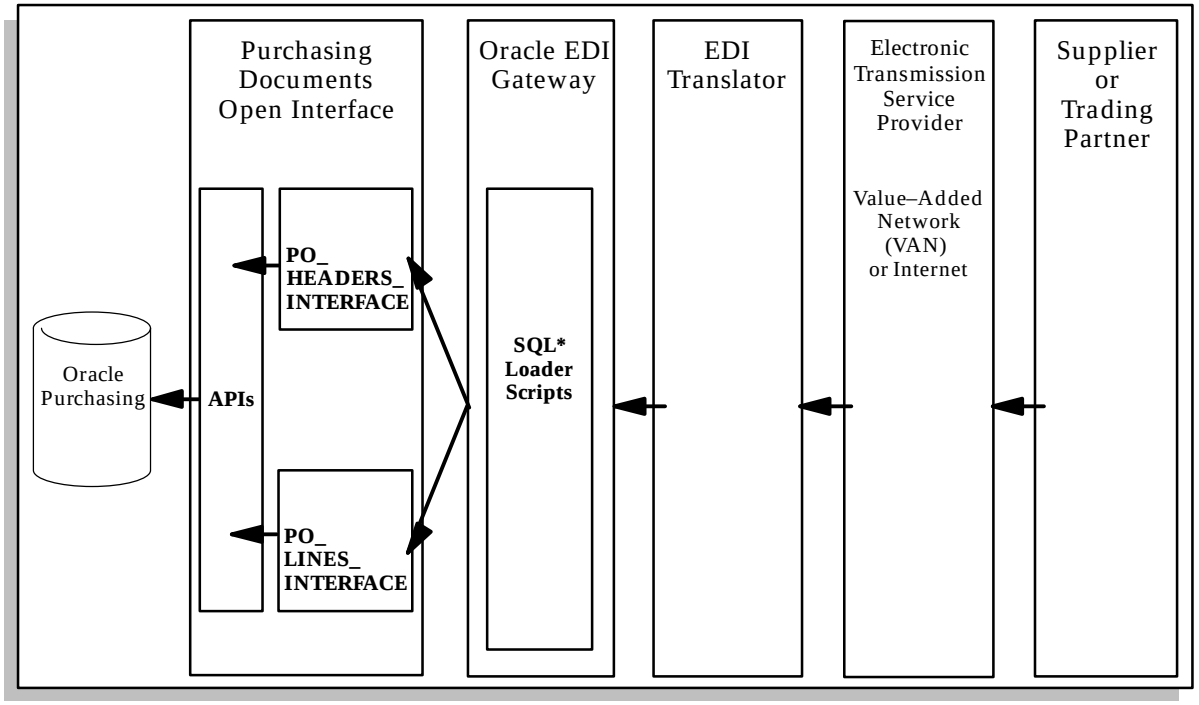
Purchasing Documents Open Interface

You can automatically import price/sales catalog information and responses to requests for quotations (RFQs) 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 responses to RFQs, the Purchasing Documents Open Interface program converts the information in the interface tables into blanket purchase agreements or catalog quotations in Purchasing.

One way to import price/sales catalog data is through Electronic Data Interchange (EDI). Through Oracle EDI Gateway, 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/or 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 chooses an action code of Replace.

This release of 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).

Figure 6 – 2



The figure above shows the flow of price/sales catalog information and responses to RFQs from the supplier or trading partner, to Oracle EDI 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 EDI Gateway, then the supplier must convert the price/sales catalog information or response to RFQ into a flat file using an EDI translator according to the EDI output 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 and shipment information.

In the Parameters window of the EDI Catalog Inbound program (or EDI Response to RFQ Inbound program), you specify the location 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.

The EDI inbound program and the Purchasing Documents Open Interface program are run one right after the other 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. However, 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 line or catalog quotation. The creation of items and/or sourcing rules also populates the corresponding tables (such as MTL_SYSTEM_ITEMS, ASL_ITEMS, ASL_SUPPLIERS, and ASL_DOCUMENTS).

If errors are found in the Purchasing Documents Open Interface tables, 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 program along with the failure reason(s) for each row.

Record 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." If the record is a child record (a record in the PO_LINES_INTERFACE table), the program will also flag the corresponding PO_HEADERS_INTERFACE record as "Not Processable" if it is not already flagged so. The program continues to

process the child records for errors, even after the first error is found, and records them in the PO_INTERFACE_ERRORS table.

If no processing errors are found during processing for a commit interval, the header record and all its child records are loaded into Purchasing, and then flagged as processed. (If you're importing data through EDI Gateway, you define the Commit Interval, or number of documents per interval, in the EDI Catalog Inbound or EDI Response to RFQ Inbound programs, in the Parameters window.)

If any errors are found in the PO_HEADERS_INTERFACE record or in any of its child records for a commit interval, none of the records for that commit interval are loaded into the Purchasing tables, and the program 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.
- Rolls back (that is, does not commit) any loading or updating of Purchasing tables done for all the PO_HEADERS_INTERFACE records in the commit interval.
- Begins processing the next record in the next commit interval.

A counter is incremented of the number of PO_HEADERS_INTERFACE records processed for the commit interval. When the counter reaches the commit interval you specify, the program commits the interval, resets the counter, and begins to process records in the next commit interval.

Types of Flat Files

If you are using EDI Gateway to import the data (or flat files) into the Purchasing Documents Open Interface, your supplier can send you flat files with two kinds of action codes (in the ACTION column): 'ORIGINAL' or 'REPLACE'. A file with an action code of ORIGINAL is one in which all the catalog information is new to your system. 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 program replaces these documents by doing the following:

- First, it looks for documents that have the same vendor (supplier) document number as the replacement documents. (A vendor document number is a field that can be specified in the

flat file that the supplier sends and that is passed into Purchasing.)

- Next, among those documents with matching supplier document numbers, 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) and creates new documents with the new price/sales catalog information, within the original effectivity dates.

Setting Up the Purchasing Documents Open Interface

If you want to import supplier catalog information into the Purchasing Documents Open Interface through EDI, you need to install and set up EDI Gateway for your organization. You also need to define your supplier as a trading partner, enable the EDI Catalog Inbound transaction for that partner, and set up the appropriate code conversions.

Also make sure that default category sets are properly set up for both Purchasing and Inventory in the Category Sets window. Also allow updating of the item master, if you want to update the item master. See: Receiving Price/Sales Catalog Information Electronically, *Oracle Purchasing User's Guide* for more information.

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

The import programs window in EDI Gateway initiates both the EDI Catalog Inbound program (or EDI Response to RFQ Inbound program) and the Purchasing Documents Open Interface program. 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 data into the PO_HEADERS_INTERFACE and PO_LINES_INTERFACE tables.

Note: It is recommended that you set the Commit Interval in the Parameters window to 1 so that the Purchasing Documents Open Interface program validates only one header–level record at a time (per commit interval). That way, if an error occurs, only one header–level record is rolled back.

Note: 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.

See Also

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

Inbound Response to Request for Quote (843/QUOTES), *Oracle EDI Gateway User's Guide*

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

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 Purchasing can derive (or default) some of the data into the Purchasing tables. Most of the columns in these tables correspond with columns in the PO_HEADERS and PO_LINES tables in Purchasing.

Most of the column descriptions below 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.

Not all columns described below are currently used 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 program itself requires, whether the data is imported through EDI Gateway or a program you write. The following definitions are used:

Required

The Purchasing Documents Open Interface program requires these values at a minimum, whether they are imported through a program you write or through EDI Gateway. For example, the Purchasing Documents Open Interface program requires a value for the column INTERFACE_HEADER_ID, and EDI Gateway provides a value automatically.

Derived and/or Defaulted

The Purchasing Documents Open Interface program is capable of deriving or defaulting 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 initial release, the Purchasing Documents Open Interface program 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 Technical Reference Manual, Release 11

Purchasing Documents Headers Table Description

The following graphic describes the PO_HEADERS_INTERFACE table.

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
INTERFACE_HEADER_ID	Number	✓			
BATCH_ID	Number			✓	
INTERFACE_SOURCE_CODE	Varchar2				✓
PROCESS_CODE	Varchar2			✓	
ACTION	Varchar2	✓			
GROUP_CODE	Varchar2				✓
ORG_ID	Number		✓	✓	
DOCUMENT_TYPE_CODE	Varchar2			✓	
DOCUMENT_SUBTYPE	Varchar2			✓	
DOCUMENT_NUM	Varchar2		✓	✓	
PO_HEADER_ID	Number		✓	✓	
RELEASE_NUM	Number			✓	
PO_RELEASE_ID	Number				✓
RELEASE_DATE	Date				✓
CURRENCY_CODE	Varchar2		✓	✓	
RATE_TYPE	Varchar2			✓	
RATE_TYPE_CODE	Varchar2		✓	✓	
RATE_DATE	Date		✓	✓	
RATE	Number		✓	✓	
AGENT_NAME	Varchar2			✓	
AGENT_ID	Number		✓	✓	

Table 6 – 5 Purchasing Documents Open Interface (Headers) (Page 1 of 5)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
VENDOR_NAME	Varchar2			✓	
VENDOR_ID	Number	✓	✓		
VENDOR_SITE_CODE	Varchar2	✓			
VENDOR_SITE_ID	Number	✓	✓		
VENDOR_CONTACT	Varchar2			✓	
VENDOR_CONTACT_ID	Number		✓	✓	
SHIP_TO_LOCATION	Varchar2			✓	
SHIP_TO_LOCATION_ID	Number		✓	✓	
BILL_TO_LOCATION	Varchar2			✓	
BILL_TO_LOCATION_ID	Number		✓	✓	
PAYMENT_TERMS	Varchar2			✓	
TERMS_ID	Number		✓	✓	
FREIGHT_CARRIER	Varchar2		✓	✓	
FOB	Varchar2		✓	✓	
FREIGHT_TERMS	Varchar2		✓	✓	
APPROVAL_STATUS	Varchar2			✓	
APPROVED_DATE	Date		✓		
REVISED_DATE	Date				✓
REVISION_NUM	Number			✓	
NOTE_TO_VENDOR	Varchar2				✓
NOTE_TO_RECEIVER	Varchar2				✓
CONFIRMING_ORDER_FLAG	Varchar2			✓	
COMMENTS	Varchar2			✓	

Table 6 – 5 Purchasing Documents Open Interface (Headers) (Page 2 of 5)

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

Table 6 – 5 Purchasing Documents Open Interface (Headers) (Page 3 of 5)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
USSGL_TRANSACTION_CODE	Varchar2			✓	
ATTRIBUTE_CATEGORY	Varchar2			✓	
ATTRIBUTE1	Varchar2			✓	
ATTRIBUTE2	Varchar2			✓	
ATTRIBUTE3	Varchar2			✓	
ATTRIBUTE4	Varchar2			✓	
ATTRIBUTE5	Varchar2			✓	
ATTRIBUTE6	Varchar2			✓	
ATTRIBUTE7	Varchar2			✓	
ATTRIBUTE8	Varchar2			✓	
ATTRIBUTE9	Varchar2			✓	
ATTRIBUTE10	Varchar2			✓	
ATTRIBUTE11	Varchar2			✓	
ATTRIBUTE12	Varchar2			✓	
ATTRIBUTE13	Varchar2			✓	
ATTRIBUTE14	Varchar2			✓	
ATTRIBUTE15	Varchar2			✓	
CREATION_DATE	Date		✓	✓	
CREATED_BY	Number		✓	✓	
LAST_UPDATE_DATE	Date		✓	✓	
LAST_UPDATED_BY	Number		✓	✓	
LAST_UPDATE_LOGIN	Number		✓	✓	
REQUEST_ID	Number		✓	✓	

Table 6 – 5 Purchasing Documents Open Interface (Headers) (Page 4 of 5)

PO_HEADERS_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
PROGRAM_APPLICATION_ID	Number		✓	✓	
PROGRAM_ID	Number		✓	✓	
PROGRAM_UPDATE_DATE	Date		✓	✓	
REFERENCE_NUM	Varchar2			✓	
LOAD_SOURCING_RULES_FLAG	Varchar2			✓	
VENDOR_NUM	Varchar2			✓	
FROM_RFQ_NUM	Varchar2			✓	
WF_GROUP_ID	Number			✓	

Table 6 – 5 Purchasing Documents Open Interface (Headers) (Page 5 of 5)

Following is a description of some of the columns in the PO_HEADERS_INTERFACE table. Remaining column descriptions can be found in the *Oracle Purchasing Technical Reference Manual, Release 11*.

INTERFACE_HEADER_ID **Required**

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

BATCH_ID **Optional**

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

INTERFACE_SOURCE_CODE **Reserved for Future Use**

This column identifies the source (for example, EDI 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', or 'REJECTED'. A PENDING transaction has not yet been processed; a FINISHED transaction has been successfully processed; a REJECTED transaction contains an error which shows up in the Purchasing Interface Errors Report.

When you import price/sales catalog information through EDI Gateway, EDI Gateway defaults a value of 'PENDING' in this column automatically; then the Purchasing Documents Open Interface program sets the value to 'ACCEPTED' or 'REJECTED'.

ACTION **Required**

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

GROUP_CODE **Reserved for Future Use**

This column indicates an identifier for the batch being imported.

LOAD_SOURCING_RULES_FLAG **Optional**

This column indicates whether to create sourcing rules with the purchasing document. If you are using EDI Gateway to import the purchasing documents, you choose this option in the EDI Catalog Inbound program or EDI Response to RFQ Inbound program Parameters window.

Purchasing Documents Lines Table Description

The following graphic describes the PO_LINES_INTERFACE table.

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
INTERFACE_LINE_ID	Number	✓			
INTERFACE_HEADER_ID	Number	✓			
ACTION	Varchar2				✓
GROUP_CODE	Varchar2				✓
LINE_NUM	Number		✓	✓	
PO_LINE_ID	Number		✓	✓	
SHIPMENT_NUM	Number		✓	✓	
LINE_LOCATION_ID	Number		✓	✓	
SHIPMENT_TYPE	Varchar2		✓	✓	
REQUISITION_LINE_ID	Number				✓
DOCUMENT_NUM	Number				✓
RELEASE_NUM	Number				✓
PO_HEADER_ID	Number		✓	✓	
PO_RELEASE_ID	Number				✓
SOURCE_SHIPMENT_ID	Number				✓
CONTRACT_NUM	Varchar2				✓
LINE_TYPE	Varchar2			✓	
LINE_TYPE_ID	Number		✓	✓	
ITEM	Varchar2	*conditionally*			
ITEM_ID	Number		✓	✓	
ITEM_REVISION	Varchar2			✓	

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 1 of 7)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
CATEGORY	Varchar2			✓	
CATEGORY_ID	Number		✓	✓	
ITEM_DESCRIPTION	Varchar2	*conditionally*	✓		
VENDOR_PRODUCT_NUM	Varchar2		✓	✓	
UOM_CODE	Varchar2		✓	✓	
UNIT_OF_MEASURE	Varchar2		✓	✓	
QUANTITY	Number		✓	✓	
COMMITTED_AMOUNT	Number			✓	
MIN_ORDER_QUANTITY	Number			✓	
MAX_ORDER_QUANTITY	Number			✓	
UNIT_PRICE	Number		✓	✓	
LIST_PRICE_PER_UNIT	Number		✓	✓	
MARKET_PRICE	Number		✓	✓	
ALLOW_PRICE_OVERRIDE_FLAG	Varchar2		✓	✓	
NOT_TO_EXCEED_PRICE	Number			✓	
NEGOTIATED_BY_PREPARER_FLAG	Varchar2		✓	✓	
UN_NUMBER	Varchar2			✓	
UN_NUMBER_ID	Number		✓	✓	
HAZARD_CLASS	Varchar2			✓	
HAZARD_CLASS_ID	Number		✓	✓	
NOTE_TO_VENDOR	Varchar2				✓
TRANSACTION_REASON_CODE	Varchar2				✓
TAXABLE_FLAG	Varchar2		✓	✓	

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 2 of 7)

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

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 3 of 7)

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

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 4 of 7)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
LINE_ATTRIBUTE_CATEGORY_LINES	Varchar2			✓	
LINE_ATTRIBUTE1	Varchar2			✓	
LINE_ATTRIBUTE2	Varchar2			✓	
LINE_ATTRIBUTE3	Varchar2			✓	
LINE_ATTRIBUTE4	Varchar2			✓	
LINE_ATTRIBUTE5	Varchar2			✓	
LINE_ATTRIBUTE6	Varchar2			✓	
LINE_ATTRIBUTE7	Varchar2			✓	
LINE_ATTRIBUTE8	Varchar2			✓	
LINE_ATTRIBUTE9	Varchar2			✓	
LINE_ATTRIBUTE10	Varchar2			✓	
LINE_ATTRIBUTE11	Varchar2			✓	
LINE_ATTRIBUTE12	Varchar2			✓	
LINE_ATTRIBUTE13	Varchar2			✓	
LINE_ATTRIBUTE14	Varchar2			✓	
LINE_ATTRIBUTE15	Varchar2			✓	
SHIPMENT_ATTRIBUTE_CATEGORY	Varchar2			✓	
SHIPMENT_ATTRIBUTE1	Varchar2			✓	
SHIPMENT_ATTRIBUTE2	Varchar2			✓	
SHIPMENT_ATTRIBUTE3	Varchar2			✓	
SHIPMENT_ATTRIBUTE4	Varchar2			✓	
SHIPMENT_ATTRIBUTE5	Varchar2			✓	
SHIPMENT_ATTRIBUTE6	Varchar2			✓	

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 5 of 7)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
SHIPMENT_ATTRIBUTE7	Varchar2			✓	
SHIPMENT_ATTRIBUTE8	Varchar2			✓	
SHIPMENT_ATTRIBUTE9	Varchar2			✓	
SHIPMENT_ATTRIBUTE10	Varchar2			✓	
SHIPMENT_ATTRIBUTE11	Varchar2			✓	
SHIPMENT_ATTRIBUTE12	Varchar2			✓	
SHIPMENT_ATTRIBUTE13	Varchar2			✓	
SHIPMENT_ATTRIBUTE14	Varchar2			✓	
SHIPMENT_ATTRIBUTE15	Varchar2			✓	
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		✓	✓	
ORGANIZATION_ID	Number		✓	✓	
ITEM_ATTRIBUTE_CATEGORY	Varchar2			✓	
ITEM_ATTRIBUTE1	Varchar2			✓	
ITEM_ATTRIBUTE2	Varchar2			✓	
ITEM_ATTRIBUTE3	Varchar2			✓	

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 6 of 7)

PO_LINES_INTERFACE Column Name	Type	Required	Derived and/or Defaulted	Optional	Reserved for Future Use
ITEM_ATTRIBUTE4	Varchar2			✓	
ITEM_ATTRIBUTE5	Varchar2			✓	
ITEM_ATTRIBUTE6	Varchar2			✓	
ITEM_ATTRIBUTE7	Varchar2			✓	
ITEM_ATTRIBUTE8	Varchar2			✓	
ITEM_ATTRIBUTE9	Varchar2			✓	
ITEM_ATTRIBUTE10	Varchar2			✓	
ITEM_ATTRIBUTE11	Varchar2			✓	
ITEM_ATTRIBUTE12	Varchar2			✓	
ITEM_ATTRIBUTE13	Varchar2			✓	
ITEM_ATTRIBUTE14	Varchar2			✓	
ITEM_ATTRIBUTE15	Varchar2			✓	
UNIT_WEIGHT	Number			✓	
WEIGHT_UOM_CODE	Varchar2			✓	
VOLUME_UOM_CODE	Varchar2			✓	
UNIT_VOLUME	Number			✓	
TEMPLATE_ID	Number		✓	✓	
TEMPLATE_NAME	Varchar2			✓	
LINE_REFERENCE_NUM	Varchar2			✓	
SOURCING_RULE_NAME	Varchar2			✓	

Table 6 – 6 Purchasing Documents Open Interface (Lines) (Page 7 of 7)

Following is a description of some of the columns in the PO_LINES_INTERFACE table. Remaining column descriptions can be found in the *Oracle Purchasing Technical Reference Manual, Release 11*.

INTERFACE_LINE_ID **Required**

This column indicates an identifier number for the purchase order or catalog line. If you import price/sales catalog information through EDI Gateway, this identifier is provided automatically.

INTERFACE_HEADER_ID **Required**

This column indicates an identifier number for the corresponding purchase order header. If you import price/sales catalog information through EDI Gateway, this identifier is provided automatically.

GROUP_CODE **Reserved for Future Use**

This column indicates an identifier for the batch being imported.

SOURCING_RULE_NAME **Optional**

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

Minimally Required Data

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

- INTERFACE_HEADER_ID – Enter a unique identifier for the record in the PO_HEADERS_INTERFACE table. If you're importing data through EDI Gateway, EDI Gateway generates a value in this column automatically.
- ACTION – This column can have one of two values: ORIGINAL to create a new catalog, or REPLACE to replace an existing catalog.
- VENDOR_ID or VENDOR_NAME – Enter the supplier for the document. Make sure the supplier is also set up as a trading partner in the EDI Gateway application, if you're importing data through EDI Gateway.

Additionally, you may have to enter values for other conditionally required columns in the PO_HEADERS_INTERFACE table. *For example*:

- VENDOR_SITE_ID or VENDOR_SITE_CODE – Enter the supplier site for the document. If the supplier has more than one site then the Purchasing Documents Open interface cannot

default a site. The site needs to be populated in the interface table; it also needs to be set up in EDI Gateway, if you're importing data through EDI Gateway.

- EFFECTIVE_DATE, EXPIRATION_DATE and VENDOR_DOC_NUM – All of these fields must be populated when replacing an existing purchasing document. The values are used to locate the old catalog and expire it.

You must always enter values for the following required columns when you load rows into the PO_LINES_INTERFACE table; if you're importing data through EDI Gateway, EDI Gateway generates values in these columns automatically:

- INTERFACE_HEADER_ID – Enter the unique identifier of the header record to which this line belongs.
- INTERFACE_LINE_ID – Enter the unique identifier of the record in the PO_LINES_INTERFACE table.

Additionally, you may have to enter values for other conditionally required columns in the PO_LINES_INTERFACE table. *For example*:

- EFFECTIVE_DATE and EXPIRATION_DATE – If sourcing rules are to be created, then you need to provide values for these columns.
- ITEM and ITEM_DESCRIPTION – If you want to create items in the item master then you need to supply the item information.

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.

Note: ITEM_DESCRIPTION cannot be derived or defaulted if you are creating a new item.

The Purchasing Documents Open Interface program 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 the normal API defaulting scheme. (Derivation refers to deriving a full value from a partial value given; defaulting refers to using a default value in Purchasing.) 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 program 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 Items (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 program does not validate those columns described as ”Reserved for Future Use” on the previous pages.

Standard Validation

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

Other Validation

The Purchasing Documents Open Interface program performs the same rigorous 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 CURRENCY_RATE_TYPE does not need to be defined but contains a value, an error will be signaled.

Resolving Failed Purchasing 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.

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 program 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:

Error Message	Meaning
PO_PDOI_AMT_LIMIT_LT_AGREED	Amount Limit (VALUE= &AMOUNT_LIMIT) is less than Amount Agreed (VALUE=&VALUE).

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 1 of 7)

Error Message	Meaning
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).
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.

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 2 of 7)

Error Message	Meaning
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.
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.

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 3 of 7)

Error Message	Meaning
PO_PDOI_INVALID_NUM_OF_LINES	&COLUMN_NAMEThere 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.
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.

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 4 of 7)

Error Message	Meaning
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	Vendor Contact (VALUE=&VALUE) is not an active and valid contact for the specified vendor site.
PO_PDOI_INVALID_VENDOR	Vendor (VALUE=&VALUE) specified is invalid or inactive.
PO_PDOI_INVALID_VENDOR_SITE	Vendor Site (VALUE=&VALUE) is not an active and valid purchasing vendor 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.

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 5 of 7)

Error Message	Meaning
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	Rate type, rate_date and rate must be null.
PO_PDOI_RULE_NAME_UNIQ	Rule Name (VALUE= &VALUE) and Item Id (ID =&VALUE) should be unique in mrp_sourcing_rules table.
PO_PDOI_SHIPMENT_NUM_UNIQUE	Shipment Num must have a unique value. &VALUE already exists.
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.

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 6 of 7)

Error Message	Meaning
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 Oracle 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.

Table 6 – 7 Purchasing Documents Open Interface Error Messages (Page 7 of 7)

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

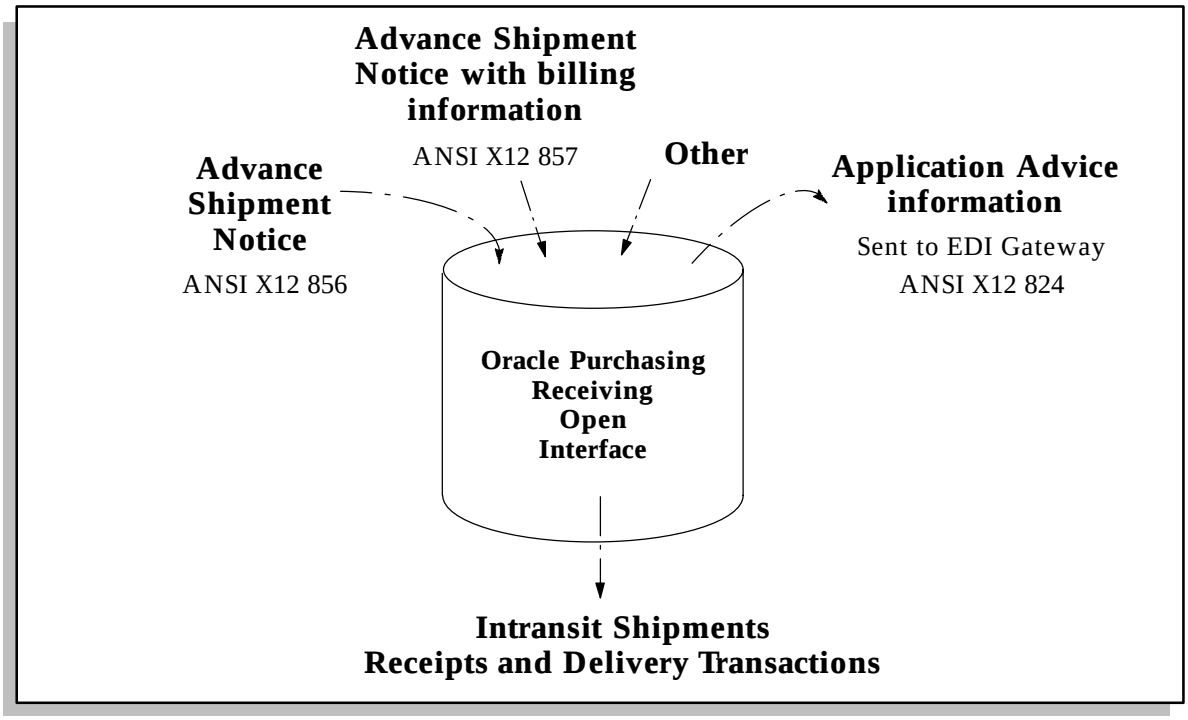
Purchasing Interface Errors Report, *Oracle Purchasing User's Guide.*

Receiving Open Interface

You can automatically import receipt information from other Oracle Applications or your existing non–Oracle systems using the Receiving Open Interface. This interface lets you integrate Oracle Purchasing quickly with new or existing applications. For example, you can easily load barcoded and other receiving information from scanners and radio frequency devices, and the Receiving Open Interface maintains the integrity of the new data as well as the receipt data already in Purchasing. Advance Shipment Notices (ASNs) sent from suppliers are also validated in the Receiving Open Interface.

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

Figure 6 – 3



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

After the inbound flat file is loaded into the RCV_HEADERS_INTERFACE and RCV_TRANSACTIONS_INTERFACE tables, the Receiving Open Interface 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'.

First, the Receiving Open Interface performs header–level validations. If no fatal errors are detected at the header level, the Receiving Open Interface selects all the lines associated with each header and performs line–level validations.

If no fatal errors are detected at the header level, at least one line was successfully validated (the profile option *RCV: Fail All ASN Lines if One Line Fails* is set to 'No'), and TEST_FLAG is set to anything but 'Y', the Receiving Open Interface does the following:

- Populates the RCV_SHIPMENT_HEADERS table in Purchasing with the header information.
- Populates the RCV_SHIPMENT_LINES table in Purchasing.
- Populates the RCV_TRANSACTIONS table in Purchasing 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 table MTL_SUPPLY.

If errors are detected, the Receiving Open Interface populates the PO_INTERFACE_ERRORS table and the outbound Application Advice EDI Gateway interface tables. A separate process downloads the contents of the outbound Application Advice EDI Gateway 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.

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 EDI Gateway interface tables with an error. (The Application Advice tables are populated when you run the Receiving Interface Errors Report.)

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 EDI 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, it cascades up to the over–receipt tolerance. For example, if there are 10 purchase order shipment lines of 100 units each and the Over Receipt Quantity Tolerance is 10%, the Receiving Open Interface can consume 10 more units for the last shipment line. If the ASN total quantity is 1,011 (at the last shipment line, 1 more item needs to be consumed), Purchasing looks at how the Over Receipt Quantity Tolerance is set in the Receiving Controls window. If the shipment exceeds the tolerance for all open shipments and the Over Receipt Quantity Action code is set to Reject, then Purchasing rejects the ASN line and creates an error in the PO_INTERFACE_ERRORS table.

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 Over Receipt Quantity Action code 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: Profile Options in Purchasing, *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 EDI Gateway. See: *Oracle EDI 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.

Inserting into the Receiving Open Interface Table

You load receipt data from your source system or EDI 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 EDI Gateway or a program you write. For example, HEADER_INTERFACE_ID is a required column; however, when receiving ASNs from suppliers through EDI Gateway, EDI 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 Maintain 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. A Derived column marked with a check (✓*) indicates that the Receiving Transaction Processor inserts values into these columns automatically, so you should not insert your own values.

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
HEADER_INTERFACE_ID	Number	✓			
GROUP_ID	Number	✓			
EDI_CONTROL_NUM	Varchar2			✓	
PROCESSING_STATUS_CODE	Varchar2	✓			
RECEIPT_SOURCE_CODE	Varchar2	✓			
ASN_TYPE	Varchar2	*conditionally*			
TRANSACTION_TYPE	Varchar2	✓			
AUTO_TRANSACT_CODE	Varchar2	*conditionally*			
TEST_FLAG	Varchar2			✓	
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number			✓	
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
NOTICE_CREATION_DATE	Date			✓	
SHIPMENT_NUM	Varchar2	*conditionally*			
RECEIPT_NUM	Varchar2	*conditionally*			
RECEIPT_HEADER_ID	Number		*conditionally*		

Table 6 – 8 Receiving Open Interface (Headers) (Page 1 of 4)

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
VENDOR_NAME	Varchar2	✓	*conditionally*		
VENDOR_NUM	Varchar2				
VENDOR_ID	Number				
VENDOR_SITE_CODE	Varchar2		*conditionally*	✓	
VENDOR_SITE_ID	Number				
FROM_ORGANIZATION_CODE	Varchar2				✓
FROM_ORGANIZATION_ID	Number				
SHIP_TO_ORGANIZATION_CODE	Varchar2	*conditionally*	*conditionally*		
SHIP_TO_ORGANIZATION_ID	Number				
LOCATION_CODE	Varchar2		*conditionally*	✓	
LOCATION_ID	Number				
BILL_OF_LADING	Varchar2			✓	
PACKING_SLIP	Varchar2			✓	
SHIPPED_DATE	Date	*conditionally*			
FREIGHT_CARRIER_CODE	Varchar2			✓	
EXPECTED_RECEIPT_DATE	Date			✓	
RECEIVER_ID	Number				✓
NUM_OF_CONTAINERS	Number			✓	
WAYBILL_AIRBILL_NUM	Varchar2			✓	
COMMENTS	Varchar2			✓	
GROSS_WEIGHT	Number			✓	
GROSS_WEIGHT_UOM_CODE	Varchar2			✓	
NET_WEIGHT	Number			✓	
NET_WEIGHT_UOM_CODE	Varchar2			✓	
TAR_WEIGHT	Number			✓	
TAR_WEIGHT_UOM_CODE	Varchar2			✓	

Table 6 – 8 Receiving Open Interface (Headers) (Page 2 of 4)

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
PACKAGING_CODE	Varchar2			✓	
CARRIER_METHOD	Varchar2			✓	
CARRIER_EQUIPMENT	Varchar2			✓	
SPECIAL_HANDLING_CODE	Varchar2			✓	
HAZARD_CODE	Varchar2			✓	
HAZARD_CLASS	Varchar2			✓	
HAZARD_DESCRIPTION	Varchar2			✓	
FREIGHT_TERMS	Varchar2			✓	
FREIGHT_BILL_NUMBER	Varchar2			✓	
INVOICE_NUM	Varchar2	*conditionally*			
INVOICE_DATE	Date	*conditionally*			
TOTAL_INVOICE_AMOUNT	Number	*conditionally*			
TAX_NAME	Varchar2			✓	
TAX_AMOUNT	Number			✓	
FREIGHT_AMOUNT	Number			✓	
CURRENCY_CODE	Varchar2			✓	
CONVERSION_RATE	Number			✓	
CONVERSION_RATE_TYPE	Varchar2			✓	
CONVERSION_RATE_DATE	Date			✓	
PAYMENT_TERMS_NAME	Varchar2		*conditionally*	✓	
PAYMENT_TERMS_ID	Number				
ATTRIBUTE_CATEGORY	Varchar2			✓	
ATTRIBUTE1	Varchar2			✓	
ATTRIBUTE2	Varchar2			✓	

Table 6 – 8 Receiving Open Interface (Headers) (Page 3 of 4)

RCV_HEADERS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
ATTRIBUTE3	Varchar2			✓	
ATTRIBUTE4	Varchar2			✓	
ATTRIBUTE5	Varchar2			✓	
ATTRIBUTE6	Varchar2			✓	
ATTRIBUTE7	Varchar2			✓	
ATTRIBUTE8	Varchar2			✓	
ATTRIBUTE9	Varchar2			✓	
ATTRIBUTE10	Varchar2			✓	
ATTRIBUTE11	Varchar2			✓	
ATTRIBUTE12	Varchar2			✓	
ATTRIBUTE13	Varchar2			✓	
ATTRIBUTE14	Varchar2			✓	
ATTRIBUTE15	Varchar2			✓	
USSGL_TRANSACTION_CODE	Varchar2			✓	
EMPLOYEE_NAME	Varchar2	*conditionally*	*conditionally*		
EMPLOYEE_ID	Number				
INVOICE_STATUS_CODE	Varchar2			✓	
VALIDATION_FLAG	Varchar2	✓			
REQUEST_ID	Number		✓ *		
PROCESSING_REQUEST_ID	Number		✓ *		

Table 6 – 8 Receiving Open Interface (Headers) (Page 4 of 4)

Receiving Transactions Interface Table Description

The following graphic describes the receiving transactions interface table. A Derived column marked with a check (✓*) indicates that the Receiving Transaction Processor inserts values into these columns automatically, so you should not insert your own values.

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
INTERFACE_TRANSACTION_ID	Number	✓			
GROUP_ID	Number	✓			
LAST_UPDATE_DATE	Date	✓			
LAST_UPDATED_BY	Number	✓			
CREATION_DATE	Date	✓			
CREATED_BY	Number	✓			
LAST_UPDATE_LOGIN	Number			✓	
REQUEST_ID	Number				✓
PROGRAM_APPLICATION_ID	Number				✓
PROGRAM_ID	Number				✓
PROGRAM_UPDATE_DATE	Date				✓
TRANSACTION_TYPE	Varchar2	✓			
TRANSACTION_DATE	Date	✓			
PROCESSING_STATUS_CODE	Varchar2	✓			
PROCESSING_MODE_CODE	Varchar2	✓			
PROCESSING_REQUEST_ID	Number		✓ *		
TRANSACTION_STATUS_CODE	Varchar2	✓			
CATEGORY_ID	Number	*conditionally*	*conditionally*		
ITEM_CATEGORY	Varchar2				

Table 6 – 9 Receiving Open Interface (Transactions) (Page 1 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
QUANTITY	Number	✓			
UNIT_OF_MEASURE	Varchar2	✓			
INTERFACE_SOURCE_CODE	Varchar2			✓	
INTERFACE_SOURCE_LINE_ID	Number				✓
INV_TRANSACTION_ID	Number				✓
ITEM_ID	Number	*conditionally*	*conditionally*		
ITEM_NUM	Varchar2				
ITEM_DESCRIPTION	Varchar2	✓			
ITEM_REVISION	Varchar2	*conditionally*	*conditionally*		
UOM_CODE	Varchar2				✓
EMPLOYEE_ID	Number	*conditionally*	*conditionally*		
AUTO_TRANSACT_CODE	Varchar2	✓			
SHIPMENT_HEADER_ID	Number				✓
SHIPMENT_LINE_ID	Number				✓
SHIP_TO_LOCATION_ID	Number	*conditionally*	*conditionally*		
SHIP_TO_LOCATION_CODE	Varchar2				
PRIMARY_QUANTITY	Number				✓
PRIMARY_UNIT_OF_MEASURE	Varchar2				✓
RECEIPT_SOURCE_CODE	Varchar2	✓			
VENDOR_ID	Number	✓	*conditionally*		
VENDOR_NUM	Varchar2				
VENDOR_NAME	Varchar2				
VENDOR_SITE_ID	Number		✓	✓	
VENDOR_SITE_CODE	Varchar2				
FROM_ORGANIZATION_ID	Number				✓

Table 6 – 9 Receiving Open Interface (Transactions) (Page 2 of 9)

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

Table 6 – 9 Receiving Open Interface (Transactions) (Page 3 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
RECEIPT_EXCEPTION_FLAG	Varchar2				✓
ACCRUAL_STATUS_CODE	Varchar2				✓
INSPECTION_STATUS_CODE	Varchar2				✓
INSPECTION_QUALITY_CODE	Varchar2				✓
DESTINATION_TYPE_CODE	Varchar2		*conditionally*	✓	
SUBINVENTORY	Varchar2	*conditionally*	*conditionally*		
WIP_ENTITY_ID	Number				✓
WIP_LINE_ID	Number				✓
DEPARTMENT_CODE	Varchar2				✓
WIP_REPETITIVE_SCHEDULE_ID	Number				✓
WIP_OPERATION_SEQ_NUM	Number				✓
WIP_RESOURCE_SEQ_NUM	Number				✓
BOM_RESOURCE_ID	Number				✓
SHIPMENT_NUM	Varchar2				✓
FREIGHT_CARRIER_CODE	Varchar2		*conditionally*		
BILL_OF_LADING	Varchar2		*conditionally*		
PACKING_SLIP	Varchar2			✓	
SHIPPED_DATE	Date				✓
EXPECTED_RECEIPT_DATE	Date	*conditionally*	*conditionally*		
ACTUAL_COST	Number			✓	
TRANSFER_COST	Number			✓	
TRANSPORTATION_COST	Number			✓	
TRANSPORTATION_ACCOUNT_ID	Number			✓	

Table 6 – 9 Receiving Open Interface (Transactions) (Page 4 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
NUM_OF_CONTAINERS	Number			✓	
WAYBILL_AIRBILL_NUM	Varchar2			✓	
VENDOR_ITEM_NUM	Varchar2	*conditionally*	*conditionally*		
VENDOR_LOT_NUM	Varchar2				✓
RMA_REFERENCE	Varchar2			✓	
COMMENTS	Varchar2			✓	
ATTRIBUTE_CATEGORY	Varchar2			✓	
ATTRIBUTE1	Varchar2			✓	
ATTRIBUTE2	Varchar2			✓	
ATTRIBUTE3	Varchar2			✓	
ATTRIBUTE4	Varchar2			✓	
ATTRIBUTE5	Varchar2			✓	
ATTRIBUTE6	Varchar2			✓	
ATTRIBUTE7	Varchar2			✓	
ATTRIBUTE8	Varchar2			✓	
ATTRIBUTE9	Varchar2			✓	
ATTRIBUTE10	Varchar2			✓	
ATTRIBUTE11	Varchar2			✓	
ATTRIBUTE12	Varchar2			✓	
ATTRIBUTE13	Varchar2			✓	
ATTRIBUTE14	Varchar2			✓	
ATTRIBUTE15	Varchar2			✓	
SHIP_HEAD_ATTRIBUTE_CATEGORY	Varchar2				✓

Table 6 – 9 Receiving Open Interface (Transactions) (Page 5 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
SHIP_HEAD_ATTRIBUTE1	Varchar2				✓
SHIP_HEAD_ATTRIBUTE2	Varchar2				✓
SHIP_HEAD_ATTRIBUTE3	Varchar2				✓
SHIP_HEAD_ATTRIBUTE4	Varchar2				✓
SHIP_HEAD_ATTRIBUTE5	Varchar2				✓
SHIP_HEAD_ATTRIBUTE6	Varchar2				✓
SHIP_HEAD_ATTRIBUTE7	Varchar2				✓
SHIP_HEAD_ATTRIBUTE8	Varchar2				✓
SHIP_HEAD_ATTRIBUTE9	Varchar2				✓
SHIP_HEAD_ATTRIBUTE10	Varchar2				✓
SHIP_HEAD_ATTRIBUTE11	Varchar2				✓
SHIP_HEAD_ATTRIBUTE12	Varchar2				✓
SHIP_HEAD_ATTRIBUTE13	Varchar2				✓
SHIP_HEAD_ATTRIBUTE14	Varchar2				✓
SHIP_HEAD_ATTRIBUTE15	Varchar2				✓
SHIP_LINE_ATTRIBUTE_CATEGORY	Varchar2				✓
SHIP_LINE_ATTRIBUTE1	Varchar2				✓
SHIP_LINE_ATTRIBUTE2	Varchar2				✓
SHIP_LINE_ATTRIBUTE3	Varchar2				✓
SHIP_LINE_ATTRIBUTE4	Varchar2				✓
SHIP_LINE_ATTRIBUTE5	Varchar2				✓
SHIP_LINE_ATTRIBUTE6	Varchar2				✓
SHIP_LINE_ATTRIBUTE7	Varchar2				✓

Table 6 – 9 Receiving Open Interface (Transactions) (Page 6 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
SHIP_LINE_ATTRIBUTE8	Varchar2				✓
SHIP_LINE_ATTRIBUTE9	Varchar2				✓
SHIP_LINE_ATTRIBUTE10	Varchar2				✓
SHIP_LINE_ATTRIBUTE11	Varchar2				✓
SHIP_LINE_ATTRIBUTE12	Varchar2				✓
SHIP_LINE_ATTRIBUTE13	Varchar2				✓
SHIP_LINE_ATTRIBUTE14	Varchar2				✓
SHIP_LINE_ATTRIBUTE15	Varchar2				✓
USSGL_TRANSACTION_CODE	Varchar2				✓
GOVERNMENT_CONTEXT	Varchar2				✓
REASON_ID	Number			✓	
DESTINATION_CONTEXT	Varchar2				✓
SOURCE_DOC_QUANTITY	Number				✓
SOURCE_DOC_UNIT_OF_MEASURE	Varchar2				✓
FROM_SUBINVENTORY	Varchar2				✓
INTRANSIT_OWNING_ORG_ID	Number				✓
MOVEMENT_ID	Number				✓
USE_MTL_LOT	Number				✓
USE_MTL_SERIAL	Number				✓
TAX_NAME	Varchar2			✓	
TAX_AMOUNT	Number			✓	
NOTICE_UNIT_PRICE	Number			✓	
HEADER_INTERFACE_ID	Number	✓			

Table 6 – 9 Receiving Open Interface (Transactions) (Page 7 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
VENDOR_CUM_SHIPPED_QUANTITY	Number			✓	
TRUCK_NUM	Varchar2			✓	
CONTAINER_NUM	Varchar2			✓	
LOCATION_CODE	Varchar2		*conditionally*	✓	
LOCATION_ID	Number				
FROM_ORGANIZATION_CODE	Varchar2				✓
INTRANSIT_OWNING_ORG_CODE	Varchar2				✓
ROUTING_CODE	Varchar2				✓
ROUTING_STEP	Varchar2				✓
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			✓	
VALIDATION_FLAG	Varchar2	✓			
SUBSTITUTE_ITEM_ID	Number		*conditionally*	✓	
SUBSTITUTE_ITEM_NUM	Varchar2				
QUANTITY_SHIPPED	Number				✓
QUANTITY_INVOICED	Number				✓
REQ_NUM	Varchar2				✓
REQ_LINE_NUM	Number				✓
REQ_DISTRIBUTION_NUM	Number				✓
WIP_ENTITY_NAME	Varchar2				✓

Table 6 – 9 Receiving Open Interface (Transactions) (Page 8 of 9)

RCV_TRANSACTIONS_INTERFACE Column Name	Type	Required	Derived	Optional	Reserved for Future Use
WIP_LINE_CODE	Varchar2				✓
RESOURCE_CODE	Varchar2				✓
SHIPMENT_LINE_STATUS_CODE	Varchar2		✓ *		
BARCODE_LABEL	Varchar2			✓	
TRANSFER_PERCENTAGE	Number				✓
QA_COLLECTION_ID	Number				✓

Table 6 – 9 Receiving Open Interface (Transactions) (Page 9 of 9)

Required Data for RCV_HEADERS_INTERFACE

Required Data

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 EDI 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 supplier of the shipment. It tells the Receiving Open Interface whether the shipment is from an external supplier or an internal organization. For this initial release, 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
- CREATED_BY
- 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

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 EDI Gateway.
- 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

Required Data

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 EDI 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
- CREATED_BY
- TRANSACTION_TYPE – This column indicates the transaction purpose code. It accepts values of 'SHIP' for a standard shipment, 'RECEIVE' for a standard receipt, or 'DELIVER' for a standard receipt and delivery transaction.
- 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 Over Receipt Quantity Action code 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
- 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 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 a shipment header and shipment line(s), a receiving transaction, or a receiving and delivery transaction.

TRANSACTION_ TYPE	AUTO_TRANSACT_CODE		
	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

Table 6 – 10 Transaction Type and Transact Code Relationships (Page 1 of 1)

- RECEIPT_SOURCE_CODE – This column indicates the supplier 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 EDI 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

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 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. If at least one value is provided, the Receiving Open Interface can derive the other. 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.

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

- 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 TRANSACTION_TYPE 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 Organizations window).
- DELIVER_TO_LOCATION_CODE or DELIVER_TO_LOCATION_ID – A value is required in at least one of these columns if the AUTO_TRANSACT_CODE is 'DELIVER'.

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 Maintain 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 Maintain 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 EDI 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 11* 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 Maintain Shipments windows.

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.

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. For every transaction in the interface table that fails validation, this report lists 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*

CHAPTER

7

Oracle Quality Open Interfaces

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

Collection Import Interface

You can use Collection Import to add quality results data to the quality data repository. You can also use Collection Import 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 it 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 involves 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 used to load this data are highly dependent on the data source.

Launching the Collection Import Manager

The next step in the Collection Import process is to launch the Collection Import Manager. The Collection Import Manager is a background process. If new rows are found, Collection Import launches one or more Import Workers. These Import Workers validate the data, insert valid records into or update existing records in the quality results data repository (QA_RESULTS), and invoke actions associated with these records.

The Collection Import Interface Table can contain multiple rows. Each row specifies either that a new record be added to the quality data repository or an existing record be updated. Every time the Collection Import Manager completes a transaction, it will either update all the records that are supposed to be updated or insert into the quality data repository all those records that are specified as ”Insert”. The Collection Import Manager can only perform one *type* of transaction (updating or inserting records) every time it completes a transaction, although it can perform that transaction on multiple records (rows).

You specify what type of transaction is 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: All types of actions, except the "Display a message to the operator" action, which must be processed online, are executed. 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 and/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 can optionally delete records that you do not want to resubmit. You use the "Update Collection Import" form to accomplish this.

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

See Also

Collection Import Interface Table: page 7 – 4

Example: Collection Import SQL Script: page 7 – 12

Collection Import Manager: page 7 – 16

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.

[Collection Import Interface Table] Column Name	Datatype	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
TRANSACTION_INTERFACE_ID	Number			✓		
LAST_UPDATE_DATE	Date		✓			
LAST_UPDATED_BY	Number		✓1			
QA_LAST_UPDATED_BY	Number		✓			
QA_LAST_UPDATED_BY_NAME	VarChar2(100)			✓		
CREATION_DATE	Date		✓1			
CREATED_BY	Number		✓1			
QA_CREATED_BY	Number		✓			
QA_CREATED_BY_NAME	VarChar2(100)			✓		
LAST_UPDATE_LOGIN	Number		✓1			
REQUEST_ID	Number		✓1			
PROGRAM_APPLICATION_ID	Number		✓1			
PROGRAM_ID	Number		✓1			
PROGRAM_UPDATE_DATE	Date		✓1			
COLLECTION_ID	Number(38)			✓		
GROUP_ID	Number		✓1			
SOURCE_CODE	Varchar2(30)				✓	
SOURCE_LINE_ID	Number				✓	

Table 7 – 1 Collection Import Interface (Page 1 of 5)

[Collection Import Interface Table] Column Name	Datatype	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
PROCESS_STATUS	Number	✓				
INSERT_TYPE	Number				✓	
MATCHINGELEMENTS	VarChar2(1000)				✓	
VALIDATE_FLAG	Number				✓	
MARKER	Number		✓[1]			
ORGANIZATION_ID	Number		✓			
ORGANIZATION_CODE	VarChar2(3)	✓				
PLAN_ID	Number		✓			
PLAN_NAME	VarChar2(30)	✓				
SPEC_ID	Number		✓			
SPEC_NAME	VarChar2(30)				✓	
DEPARTMENT_ID	Number		✓			✓
DEPARTMENT	VarChar2(10)					✓
RESOURCE_ID	Number		✓			✓
RESOURCE_CODE	VarChar2(10)					✓
QUANTITY	Number					✓
ITEM_ID	Number		✓			✓
ITEM	VarChar2 (2000)					✓
UOM	VarChar2(3)					✓
REVISION	VarChar2(3)					✓
SUBINVENTORY	VarChar2(10)					✓
LOCATOR_ID	Number		✓			✓
LOCATOR	VarChar2 (2000)					✓

Table 7 – 1 Collection Import Interface (Page 2 of 5)

[Collection Import Interface Table] Column Name	Datatype	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
LOT_NUMBER	VarChar2(30)					✓
SERIAL_NUMBER	VarChar2(30)					✓
COMP_ITEM_ID	Number		✓			✓
COMP_ITEM	VarChar2 (2000)					✓
COMP_UOM	VarChar2 (3)					✓
COMP_REVISION	VarChar2 (3)					✓
COMP_SUBINVENTORY	VarChar2 (10)					✓
COMP_LOCATOR_ID	Number		✓			✓
COMP_LOCATOR	VarChar2 (2000)					✓
COMP_LOT_NUMBER	VarChar2(30)					✓
COMP_SERIAL_NUMBER	VarChar2(30)					✓
WIP_ENTITY_ID	Number		✓			✓
JOB_NAME	VarChar2(240)					✓
LINE_ID	Number		✓			✓
PRODUCTION_LINE	VarChar2(10)					✓
TO_OP_SEQ_NUM	Number					✓
FROM_OP_SEQ_NUM	Number					✓
VENDOR_ID	Number		✓			✓
VENDOR_NAME	VarChar2(80)					✓
RECEIPT_NUM	VarChar2(30)					✓
PO_HEADER_ID	Number		✓			✓
PO_NUMBER	VarChar2(20)					✓
PO_LINE_NUM	Number					✓

Table 7 – 1 Collection Import Interface (Page 3 of 5)

[Collection Import Interface Table] Column Name	Datatype	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
PO_SHIPMENT_NUM	Number					✓
CUSTOMER_ID	Number		✓			✓
CUSTOMER_NAME	VarChar2(50)					✓
SO_HEADER_ID	Number		✓			✓
SALES_ORDER	Number					✓
RESTRICT_LOCATORS_CODE	Number		✓1			✓
LOCATION_CONTROL_CODE	Number		✓1			✓
REVISION_QTY_CONTROL_CODE	Number		✓1			✓
RESTRICT_SUBINV_CODE	Number		✓1			✓
SUB_LOCATOR_TYPE	Number		✓1			✓
GEN_LOC_CTRL_CODE	Number		✓1			✓
COMP_RESTRICT_LOCATORS_CODE	Number		✓1			
COMP_LOCATION_CONTROL_CODE	Number		✓1			
COMP_REVISION_QTY_CONTROL_CODE	Number		✓1			
COMP_RESTRICT_SUBINV_CODE	Number		✓1			
COMP_SUB_LOCATOR_TYPE	Number		✓1			
COMP_GEN_LOC_CTRL_CODE	Number		✓1			
RMA_HEADER_ID	Number		✓			✓
RMA_NUMBER	Number					✓
TO_DEPARTMENT_ID	VarChar2(10)		✓			✓
TO_DEPARTMENT	Number					✓
PO_RELEASE_ID	Number			✓		✓
PO_RELEASE_NUM	Number		✓			✓

Table 7 – 1 Collection Import Interface (Page 4 of 5)

[Collection Import Interface Table] Column Name	Datatype	Required	Derived (Leave Null)	Derived or User Optional	Optional	Plan Specific
PO_TYPE_LOOKUP	VarChar2(25)		✓			✓
PROJECT_ID	Number		✓			✓
PROJECT_NUMBER	VarChar2(25)		✓			✓
STATUS	VarChar2(25)		✓			✓
TASK_ID	Number		✓			✓
TASK_NUMBER	VarChar2(25)		✓			✓
TRANSACTION_DATE	Date		✓			
CHARACTER1 through CHARACTER100	VarChar(150)					✓
[1] These columns must be left null in all circumstances.						

Table 7 – 1 Collection Import Interface (Page 5 of 5)

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 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: There may be instances where it is useful to keep status 4 (Completed) rows in the Collection Import Interface Table. You can prevent status 4 rows from being deleted, by setting the Oracle Master Scheduling/MRP and Supply Chain Planning *MRP:Debug Mode* profile option to Yes. See the *Oracle Master Scheduling/MRP User's Guide* for more details on profile options.

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 will not be rejected.

INSERT_TYPE: This field determines whether the Collection Import Manager will insert new records into the Quality data repository or update existing records in the repository. There are two values for this field:

- 1 Insert
- 2 Update

This field defaults to 1 Insert. When set to 1, or left blank, Collection Import inserts new records into the Quality data repository. When set to 2, existing records in the Quality data repository are updated.

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, that record is updated to become identical to the corresponding row in the Collection Import Interface table. However, when that record is updated to match the table row, those columns which are set to NULL in the table will not be updated in the repository. Also, an import row in the table can only match one and only one record in the repository; if the row in question has no match or 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.

Who Columns

Collection Import derives values for the standard who columns using the name of the current user:

- QA_LAST_UPDATE_BY_NAME
- QA_CREATED_BY_NAME

You can, however, insert data into these derived who columns. If you do so, the Collection Import Manager validates but does not override your data.

Optional Data

All columns that contain user–defined, reference information, and predefined collection elements are optional.

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 these two columns — CUSTOMER_ID and CUSTOMER_NAME — 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, as they are transferred directly into the results table without undergoing any 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 View

Collection Import Results Database Views are created and updated when you create and update collection plans. Collection import result database views facilitate the insertion of data into the Collection Import Interface Table. Instead of inserting data directly into the import table, you insert data into views of the table.

The Collection Import Results Database 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 import view also eliminates import 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 import view.

See Also

Collection Import Manager: page 7 – 16

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 following naming convention for collection import results database views: Q_<collection–plan–name>_IV. For example, consider the following collection plan called IMPORT with the following collection elements:

- Item
- Revision
- Lot Number
- Job Number
- To Ops Seq Number
- From Ops Seq Number
- Department
- Operator
- Defects
- Thickness
- Off Location

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;	**DataType**
TRANSACTION_INTERFACE_ID	Number
QA_LAST_UPDATED_BY_NAME	VarChar2(100)
QA_CREATED_BY_NAME	VarChar2(100)
COLLECTION_ID	Number
SOURCE_CODE	VarChar2(30
SOURCE_LINE_ID	Number
PROCESS_STATUS	Number
ORGANIZATION_CODE	VarChar2(3)
PLAN_NAME	VarChar2(30)
INSERT_TYPE	Number
MATCHING_ELEMENTS	VarChar2(1000)
SPEC_NAME	VarChar2(30)
ITEM	VarChar2 (2000)
REVISION	VarChar2(3)
LOT_NUMBER	VarChar2(30)
JOB_NAME	VarChar2(240)
FROM_OP_SEQ_NUM	Number
TO_OP_SEQ_NUM	Number
DEPARTMENT	VarChar2(10)
OPERATOR	VarChar(150)
DEFECTS	VarChar(150)
THICKNESS	VarChar(150)
OFF_LOCATION	VarChar(150)

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. If there are new rows, it launches one or more Collection Import Worker processes. You can specify the maximum number of rows you would like each worker process to handle when you launch the Collection Import Manager. Worker processes carry out the three main phases of the import process:

- Validation
- Transfer
- Error handling

The Collection Import Manager can handle an unlimited number of rows, no matter what maximum you set for each worker process. If it needs to, the Collection Import Manager will automatically launch new workers to handle more rows. For example, if you specify ten (10) to be the maximum number of rows that you would like each worker process to handle, but your interface table has 53 new records, the Collection Import Manager will automatically launch six concurrent workers, the first five handling ten rows each and the sixth handling 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 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 datatype (e.g. a value of 'abc' for pH, when pH is a number datatype)
- 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
- 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: page 7 – 4

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:

- Item
- Revision
- Lot Number
- Job Number
- To Ops Seq Number
- From Ops Seq Number
- Department
- Operator
- Defects
- Thickness
- Off Location

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;	DataType
ROW_ID	Undefined
LAST_UPDATE_DATE	Date
LAST_UPDATED_BY_ID	Number
LAST_UPDATED_BY	VarChar2(100)
LAST_UPDATE_LOGIN	Number
CREATION_DATE	Date
CREATED_BY_ID	Number
CREATED_BY	VarChar2(100)
COLLECTION_ID	Number
OCCURRENCE	Number
ORGANIZATION_ID	Number
ORGANIZATION_NAME	VarChar2(60)
PLAN_ID	Number
PLAN_NAME	VarChar2(30)
ITEM	VarChar2 (2000)
REVISION	VarChar2(3)
LOT_NUMBER	VarChar2(30)
JOB_NAME	VarChar2(240)
FROM_OP_SEQ_NUM	Number
TO_OP_SEQ_NUM	Number
DEPARTMENT	VarChar2(10)
OPERATOR	VarChar(150)
DEFECTS	VarChar(150)
THICKNESS	VarChar(150)
OFF_LOCATION	VarChar(150)

CHAPTER

8

Oracle Service Open Interfaces

This chapter contains information about the following Oracle Service open interface:

Service Request Interfaces

Oracle Service provides interfaces that enable you to import and update service requests as well as their respective statuses, severities, urgencies, owners, and problem codes. The CS_ServiceRequest_PUB package includes the following public APIs:

- CS_ServiceRequest_PUB.Create_ServiceRequest
- CS_ServiceRequest_PUB.Update_ServiceRequest
- CS_ServiceRequest_PUB.Update_Status
- CS_ServiceRequest_PUB.Update_Severity
- CS_ServiceRequest_PUB.Update_Urgency
- CS_ServiceRequest_PUB.Update_Owner
- CS_ServiceRequest_PUB.Update_Problem_Code

Features

Multi–Org

All of the APIs and procedures take advantage of the Multi–Org architecture. Whenever possible, the partitioned base tables are accessed instead of the Multi–Org views. Each interface accepts the *p_org_id* parameter as the organization ID. If you do not pass in an organization ID, the interface first checks to see whether it has been set in the session environment variable CLIENT_INFO. If it has not, the value is retrieved from the *MO: Operating Unit* profile option. If the interface cannot determine a default organization and if Multi–Org has been set up, the interface returns an error.

Flexfield Validation

The two flexfields associated with service requests are the System Items key flexfield, owned by Oracle Inventory, and the CS_INCIDENTS_ALL descriptive flexfield. Inventory items represented by the System Items flexfield can be interfaced to Oracle Service in one of three ways:

- The item's ID (p_inventory_item_id)
- A string containing a concatenation of the flexfield segments (p_inventory_item_conc_segs)

- Individual segments (p_inventory_item_segment1–20)

The interfaces retrieve the key flexfield code from the *Service: Item Flexfield (Product)* profile option, and convert and validate the segments if the ID is not passed. You can specify whether the input segments are values or hidden IDs by setting the *p_inventory_item_vals_or_id* parameter.

The interfaces also perform validation on the descriptive flexfield segments in CS_INCIDENTS_ALL (p_request_segment1–15, p_request_context). Unlike for the key flexfield, the interfaces only accept IDs for descriptive flex segments, due to flexfield API restrictions.

Oracle Self–Service Web Applications

To facilitate setting defaults for users who are also your customers, Oracle Service provides several profile options for use with Oracle Self–Service Web Applications. The interfaces for creating and updating service requests, Create_ServiceRequest and Update_ServiceRequest, accept the *p_web_entry_flag* parameter. This parameter's value indicates whether a customer has entered or updated a service request via the Web, and determines which sets of profile values are used.

Oracle Workflow

In the Service Requests window, Oracle Service performs special validation on service requests that have active workflow processes. For example, while a service request's workflow process is active, the request's owner and type cannot be updated. All of the interfaces support this validation.

The *Service: Auto Launch Workflow* profile's setting determines whether a workflow process is launched automatically when you enter a new service request in the Service Requests window. The Create_ServiceRequest interface checks this profile's setting as well, but it also considers the *p_launch_workflow* parameter. If both the profile and the parameter are set, a workflow process is launched after your imported service request is saved in the database. If the *p_web_entry_flag* is set, the *Service: Auto Launch Web Workflow* profile is checked instead of *Service: Auto Launch Workflow*.

ConText Option

If you have enabled the ConText option, whenever a column changes that is used with Oracle Service's knowledge base, the changed column's corresponding context column must be re–indexed. In Release 11, the columns used with the knowledge base are SUMMARY, PROBLEM_DESCRIPTION, and RESOLUTION_DESCRIPTION. Only published service requests will be re–indexed.

See Also

Overview of Service Requests, *Oracle Service User's Guide*

Oracle Inventory Flexfields, *Oracle Inventory User's Guide*

Service Requests, *Oracle Web Customers*

Oracle Service Profile Options, *Oracle Service User's Guide*

Overview of Oracle Workflow, *Oracle Workflow Guide*

Expert Mode Search with Oracle ConText, *Oracle Web Customers*

Values and IDs

Each of the seven interfaces accepts the responsibility application ID, responsibility ID, and user ID as optional parameters. If these IDs are not passed in, the interface gets default values from the FND_GLOBAL package and uses them to derive default values for other parameters from your profile option settings.

Also, each interface accepts both values and IDs. If you pass in both a value and an ID for the same entity, the interface uses only the ID. If you pass in only a value, the interface calls a conversion procedure to convert the value to an internal ID.

For value parameters that are stored in the database directly, without validation, the interfaces check to ensure that the passed string's length does not exceed the length of its destination column. If the check fails, the value truncates and a warning is appended to the message list.

The following table shows corresponding name and ID parameter pairs.

NAME OR NUMBER	ID
p_request_number	p_request_id
p_type_name	p_type_id
p_status_name	p_status_id
p_severity_name	p_severity_id
p_urgency_name	p_urgency_id
p_customer_name p_customer_number	p_customer_id
p_employee_name p_employee_number	p_employee_id
p_cp_ref_number	p_customer_product_id
p_rma_number	p_rma_header_id
p_inventory_item_conc_segs p_inventory_item_segment1–20	p_inventory_item_id

Table 8 – 1 Name/Number and ID Correspondences (Page 1 of 1)

For customer and employee names, numbers, and IDs: if you enter both an ID and a number, the ID takes precedence, and if you enter both a number and a name, the number takes precedence.

Value–ID Conversions

If an attribute's ID is not passed and if its value is passed as NULL, the ID is converted to NULL. For example, if *p_urgency_id* = FND_API.G_MISS_NUM and *p_urgency_name* = NULL, NULL is inserted into CS_INCIDENTS_ALL.INCIDENT_URGENCY_ID. In this case, because NULL was passed explicitly, the urgency ID will not default from the *Service: Default Service Request Urgency* profile setting.

Prerequisites

Oracle Service's interfaces expect that data such as customers and items will be defined before you create or update service requests that reference that information. Therefore, before using the interfaces, you should set up and/or activate the following parameters that Oracle Service validates:

- Request types
- Request statuses
- Request severities
- Request urgencies
- Employees (service request owners and filers)
- Customers, locations, and contacts
- Customer products in your installed base
- Serviceable items
- Problem codes
- Resolution codes

See Also

Overview of Setting Up, *Oracle Service User's Guide*

Entering New People, *Oracle Human Resources Management Systems User's Guide*

Entering Customers, *Oracle Receivables User's Guide*

Parameter Descriptions

Create_ServiceRequest

The Create_ServiceRequest interface creates a service request in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 11.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required
p_request_date	IN	DATE	
p_type_id	IN	NUMBER	
p_type_name	IN	VARCHAR2(30)	
p_status_id	IN	NUMBER	
p_status_name	IN	VARCHAR2(30)	
p_severity_id	IN	NUMBER	
p_severity_name	IN	VARCHAR2(30)	
p_urgency_id	IN	NUMBER	
p_urgency_name	IN	VARCHAR2(30)	
p_closed_date	IN	DATE	
p_owner_id	IN	NUMBER	
p_publish_flag	IN	VARCHAR2	
p_summary	IN	VARCHAR2(80)	Required
p_verify_request_flag	IN	VARCHAR2(1)	
p_filed_by_emp_flag	IN	VARCHAR2(1)	
p_customer_id	IN	NUMBER	
p_customer_number	IN	VARCHAR2(30)	
p_customer_name	IN	VARCHAR2(50)	
p_employee_id	IN	NUMBER	

Table 8 – 2 Create_ServiceRequest Parameters (Page 1 of 4)

Parameter	Usage	Type	Required?
p_employee_number	IN	VARCHAR2(30)	
p_employee_name	IN	VARCHAR2(240)	
p_contact_id	IN	NUMBER	
p_contact_name	IN	VARCHAR2(100)	
p_contact_area_code	IN	VARCHAR2(10)	
p_contact_telephone	IN	VARCHAR2(25)	
p_contact_extension	IN	VARCHAR2(20)	
p_contact_fax_area_code	IN	VARCHAR2(10)	
p_contact_fax_number	IN	VARCHAR2(25)	
p_contact_time_diff	IN	NUMBER	
p_contact_email_address	IN	VARCHAR2(240)	
p_represented_by_name	IN	VARCHAR2(100)	
p_represented_by_area_code	IN	VARCHAR2(10)	
p_represented_by_telephone	IN	VARCHAR2(25)	
p_represented_by_extension	IN	VARCHAR2(20)	
p_represented_by_fax_area_code	IN	VARCHAR2(10)	
p_represented_by_fax_number	IN	VARCHAR2(25)	
p_represented_by_time_diff	IN	NUMBER	
p_represented_by_email	IN	VARCHAR2(240)	
p_verify_cp_flag	IN	VARCHAR2(1)	
p_customer_product_id	IN	NUMBER	
p_cp_ref_number	IN	NUMBER	
p_inventory_item_id	IN	NUMBER	
p_inventory_org_id	IN	NUMBER	
p_inventory_item_conc_segs	IN	VARCHAR2	
p_inventory_item_segment1–20	IN	VARCHAR2(40)	
p_inventory_item_vals_or_ids	IN	VARCHAR2(1)	

Table 8 – 2 Create_ServiceRequest Parameters (Page 2 of 4)

Parameter	Usage	Type	Required?
p_current_serial_number	IN	VARCHAR2(30)	
p_original_order_number	IN	NUMBER	
p_purchase_order_num	IN	VARCHAR2(50)	
p_problem_description	IN	VARCHAR2(2000)	
p_problem_code	IN	VARCHAR2(30)	
p_exp_resolution_date	IN	DATE	
p_make_public_problem	IN	VARCHAR2(1)	
p_install_location	IN	VARCHAR2(40)	
p_install_customer	IN	VARCHAR2(50)	
p_install_address_line_1	IN	VARCHAR2(240)	
p_install_address_line_2	IN	VARCHAR2(240)	
p_install_address_line_3	IN	VARCHAR2(240)	
p_rma_flag	IN	VARCHAR2(1)	
p_rma_header_id	IN	NUMBER	
p_rma_number	IN	NUMBER	
p_order_type_id	IN	NUMBER	
p_web_entry_flag	IN	VARCHAR2(1)	
p_request_segment1–15	IN	VARCHAR2(150)	
p_request_context	IN	VARCHAR2(30)	
p_bill_to_site_use_id	IN	NUMBER	
p_bill_to_contact_id	IN	NUMBER	
p_bill_to_location	IN	VARCHAR2(40)	
p_bill_to_customer	IN	VARCHAR2(50)	
p_bill_address_line_1	IN	VARCHAR2(240)	
p_bill_address_line_2	IN	VARCHAR2(240)	
p_bill_address_line_3	IN	VARCHAR2(240)	
p_bill_to_contact	IN	VARCHAR2(100)	

Table 8 – 2 Create_ServiceRequest Parameters (Page 3 of 4)

Parameter	Usage	Type	Required?
p_ship_to_site_use_id	IN	NUMBER	
p_ship_to_contact_id	IN	NUMBER	
p_ship_to_location	IN	VARCHAR2(40)	
p_ship_to_customer	IN	VARCHAR2(50)	
p_ship_address_line_1	IN	VARCHAR2(240)	
p_ship_address_line_2	IN	VARCHAR2(240)	
p_ship_address_line_3	IN	VARCHAR2(240)	
p_ship_to_contact	IN	VARCHAR2(100)	
p_problem_resolution	IN	VARCHAR2(2000)	
p_resolution_code	IN	VARCHAR2(30)	
p_act_resolution_date	IN	DATE	
p_make_public_resolution	IN	VARCHAR2(1)	
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	
p_launch_workflow	IN	VARCHAR2(1)	
p_nowait	IN	VARCHAR2(1)	
p_return_status	OUT	VARCHAR2(1)	
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_request_id	OUT	NUMBER	
p_request_number	OUT	VARCHAR2(64)	
p_call_id	OUT	NUMBER	
p_itemkey	OUT	VARCHAR2(240)	
p_return_status_wkflw	OUT	VARCHAR2(1)	

Table 8 – 2 Create_ServiceRequest Parameters (Page 4 of 4)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_date Not Null **DATE**

Enter the service request date.

Default Value SYSDATE

p_type_id Not Null **NUMBER**

Enter a valid and active service request type ID. If you enter both a name and an ID, the ID is used and the name ignored. If you enter neither and the interface cannot derive a default from your profile settings, an error is returned.

Default Value *Service: Default Web Service Request Type* if you set *p_web_entry_flag*; otherwise *Service: Default Service Request Type*

p_type_name **VARCHAR2(30)**

Enter the service request type name. If you entered a value for *p_type_id*, you need not supply a name here.

p_status_id Not Null **NUMBER**

Enter a valid and active service request status ID. If you enter both a name and an ID, the ID is used and the name ignored.

Default Value 1 (open)

p_status_name **VARCHAR2(30)**

Enter the service request status name. If you entered a value for *p_status_id*, you need not supply a name here.

p_severity_id Not Null **NUMBER**

Enter a valid and active service request severity ID. If you enter both a name and an ID, the ID is used and the name ignored. If you enter

neither and the interface cannot derive a default from your profile settings, an error is returned.

Default Value *Service: Default Web Service Request Severity* if you set *p_web_entry_flag*; otherwise *Service: Default Service Request Severity*

p_severity_name **VARCHAR2(30)**

Enter the service request severity name. If you entered a value for *p_severity_id*, you need not supply a name here.

p_urgency_id **NUMBER**

Enter a valid and active service request urgency ID. If you enter both a name and an ID, the ID is used and the name ignored.

Default Value *Service: Default Web Service Request Urgency* if you set *p_web_entry_flag*; otherwise *Service: Default Service Request Urgency*

p_urgency_name **VARCHAR2(30)**

Enter the service request urgency name. If you entered a value for *p_urgency_id*, you need not supply a name here.

p_closed_date **DATE**

Enter the date this service request was closed. This date must be later than the value of *p_request_date*. If the request's status is not closed, any value entered here is ignored.

Default Value SYSDATE if the status is a closed status

p_owner_id Not Null **NUMBER**

Identify a valid and active employee ID as the service request's owner. If you do not enter a value here and the interface cannot derive a default from your profile settings, an error is returned.

Default Value *Service: Default Web Service Request Owner* if you set *p_web_entry_flag*; otherwise *Service: Default Service Request Owner*

p_publish_flag **VARCHAR2**

Indicate whether the service request is published (entered into your knowledge base), if you have enabled the ConText server option. You can set this flag to TRUE only if *Service: Publish Flag Update Allowed* is set to Yes.

p_summary Not Null **VARCHAR2(80)**

Enter a summary line.

p_verify_request_flag Not Null **VARCHAR2(1)**

Indicate whether the service request is entered in verified mode. If this flag is TRUE, the interface validates the following attributes: *customer_id*, *employee_id*, *contact_id*, *bill_to_site_use_id*, *bill_to_contact_id*, *ship_to_site_use_id*, and *ship_to_contact_id*.

Default Value FND_API.G_TRUE

p_filed_by_emp_flag Not Null **VARCHAR2(1)**

Indicate whether the service request is filed by an employee. You can set this flag to TRUE only if you also set *p_verify_request_flag*.

Default Value FND_API.G_FALSE

p_customer_id **NUMBER**

Enter the customer ID associated with the service request. This parameter is required if you set *p_verify_request_flag* but not *p_filed_by_emp_flag*.

p_customer_number **VARCHAR2(30)**

Enter the customer number associated with the service request. This parameter is ignored if you set *p_verify_request_flag*.

p_customer_name **VARCHAR2(50)**

Enter the customer name associated with the service request. This is ignored if you set *p_verify_request_flag*; it is required if you do not set *p_verify_request_flag* and if you do not use IDs. If you entered a value for *p_customer_id*, you need not supply a name here.

p_employee_id **NUMBER**

Enter the employee ID associated with the service request. This parameter is required if *p_filed_by_emp_flag* is set and ignored if that parameter is not set.

p_employee_number **VARCHAR2(30)**

Enter the employee number associated with the service request.

p_employee_name **VARCHAR2(240)**

Enter the employee name associated with the service request. If you entered a value for *p_employee_id*, you need not supply a name here.

p_contact_id **NUMBER**

Enter the customer contact ID associated with the service request. This parameter is ignored if you do not set *p_verify_request_flag*.

p_contact_name **VARCHAR2(100)**

Enter the customer contact name associated with the service request. This parameter is ignored if you set *p_verify_request_flag*. If you enter a value for *p_contact_id*, you need not supply a name here.

p_contact_time_diff **NUMBER**

Enter the time zone difference between your location and that of the customer contact. The difference must be between –24 and 24.

p_represented_by_time_diff **NUMBER**

Enter the time zone difference between your location and that of the customer representative. The difference must be between –24 and 24.

p_verify_cp_flag Not Null **VARCHAR2(1)**

Indicate whether to validate customer product IDs against your installed base. You can enter TRUE if and only if *p_verify_request_flag* is set.

p_customer_product_id Not Null **NUMBER**

Enter the unique ID of a customer product in your installed base. This parameter is required if *p_verify_cp_flag* is set; otherwise, it is ignored.

p_cp_ref_number **NUMBER**

Enter the reference number of a customer product in your installed base.

p_inventory_item_id **NUMBER**

Enter the inventory item ID that identifies your product. This corresponds to the INVENTORY_ITEM_ID column in the MTL_SYSTEM_ITEMS table. Any value you enter here is ignored if you set *p_verify_cp_flag*.

p_inventory_org_id **NUMBER**

Specify the inventory organization ID for the organization where the item is located. This is part of the unique key that identifies your inventory item and corresponds to the ORGANIZATION_ID column in the MTL_SYSTEM_ITEMS table. You must enter a value here to convert the inventory item ID.

Default Value *OE: Item Validation Organization*

p_inventory_item_conc_segs **VARCHAR2**

Enter a string consisting of a concatenation of your System Items key flexfield segments.

p_inventory_item_segment1–20 **VARCHAR2(40)**

Enter any information you want to pass to the System Items key flexfield. You can enter data for as many segments as you have enabled.

p_inventory_item_vals_or_ids **VARCHAR2(1)**

Indicate whether the input segments are values ('V') or hidden IDs ('I'). If you input values, the interface expects one value for each displayed segment. If you use IDs, the interface expects one ID for each enabled segment, regardless of whether it is displayed.

Default Value 'V'

p_current_serial_number **VARCHAR2(30)**

Enter the serial number for your item, if applicable. This parameter is ignored if you set *p_verify_cp_flag*.

p_original order_number **NUMBER**

Enter the sales order number associated with the service request. This parameter is ignored if you set *p_verify_cp_flag*.

p_purchase_order_num **VARCHAR2(50)**

Enter the purchase order number associated with the service request. This parameter is ignored if you set *p_verify_cp_flag*.

p_problem_description **VARCHAR2(2000)**

Enter the service request problem description.

p_problem_code **VARCHAR2(30)**

Enter the problem code associated with the service request.

p_exp_resolution_date **DATE**

Enter the service request's expected resolution date. This date must be after the date passed by *p_request_date.*

p_make_public_problem **VARCHAR2(1)**

Indicate whether the problem description is public.

Default Value *Service: Default Make Public Flag*

p_rma_flag Not Null **VARCHAR2(1)**

Indicate whether an RMA (return material authorization) is assigned to the service request. If this parameter is TRUE, the interface performs validation on *p_rma_header_id*. You can enter TRUE only if you also set *p_verify_request_flag*.

p_rma_header_id **NUMBER**

Enter the RMA header ID. This parameter is ignored if *p_rma_flag* is not set.

p_rma_number **NUMBER**

Enter the RMA number visible in the Returns window.

p_order_type_id **NUMBER**

Enter the ID of the RMA's order type. You must enter a value here to convert the header ID.

p_web_entry_flag Not Null **VARCHAR2(1)**

Indicate whether the service request is entered from Oracle Self–Service Web Applications.

Default Value FND_API.G_FALSE

p_request_segment1–15 **VARCHAR2(150)**

Enter any information you want to pass to the service request descriptive flexfield.

p_request_context **VARCHAR2(30)**

Enter the name of the column that defines the descriptive flexfield structure.

p_bill_to_site_use_id **NUMBER**

Enter the bill–to site ID. This parameter and *p_bill_to_contact_id* are ignored if you do not set *p_verify_request_flag*. The same applies to the corresponding ship–to parameters.

p_bill_to_location **VARCHAR2(40)**

Enter the bill–to location. This parameter, as well as the bill–to address lines, customer, and contact name, is ignored if you set *p_verify_request_flag*. The same applies to the corresponding ship–to parameters.

p_problem_resolution **VARCHAR2(2000)**

Enter the service request problem resolution.

p_resolution_code **VARCHAR2(30)**

Enter the code for the problem resolution.

p_act_resolution_date **DATE**

Enter the service request's actual resolution date. This date must be after the date passed by *p_request_date*.

p_make_public_resolution **VARCHAR2(1)**

Indicate whether the problem resolution is public.

Default Value *Service: Default Make Public Flag*

p_comments **VARCHAR2(2000)**

Enter any comments associated with the service request.

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

p_launch_workflow **VARCHAR2(1)**

Specify whether the interface will call the Workflow API to launch a workflow process for the service request.

Default Value *Service: Auto Launch Web Workflow* if you set *p_web_entry_flag*; otherwise *Service: Auto Launch Workflow*

p_nowait **VARCHAR2(1)**

If you enter TRUE, the Workflow API attempts to lock the service request record with the NOWAIT option.

Default Value FND_API.G_FALSE

Update_ServiceRequest

The Update_ServiceRequest interface updates service requests stored in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 23.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required
p_request_id	IN	NUMBER	
p_request_number	IN	VARCHAR2(64)	
p_type_id	IN	NUMBER	
p_type_name	IN	VARCHAR2(30)	
p_status_id	IN	NUMBER	
p_status_name	IN	VARCHAR2(30)	
p_severity_id	IN	NUMBER	
p_severity_name	IN	VARCHAR2(30)	
p_urgency_id	IN	NUMBER	
p_urgency_name	IN	VARCHAR2(30)	
p_closed_date	IN	DATE	
p_owner_id	IN	NUMBER	
p_publish_flag	IN	VARCHAR2	
p_summary	IN	VARCHAR2(80)	
p_verify_request_flag	IN	VARCHAR2(1)	
p_customer_id	IN	NUMBER	
p_customer_number	IN	VARCHAR2(30)	
p_customer_name	IN	VARCHAR2(50)	
p_contact_id	IN	NUMBER	

Table 8 – 3 Update ServiceRequest Parameters (Page 1 of 4)

Parameter	Usage	Type	Required?
p_contact_name	IN	VARCHAR2(100)	
p_contact_area_code	IN	VARCHAR2(10)	
p_contact_telephone	IN	VARCHAR2(25)	
p_contact_extension	IN	VARCHAR2(20)	
p_contact_fax_area_code	IN	VARCHAR2(10)	
p_contact_fax_number	IN	VARCHAR2(25)	
p_contact_time_diff	IN	NUMBER	
p_contact_email_address	IN	VARCHAR2(240)	
p_represented_by_name	IN	VARCHAR2(100)	
p_represented_by_area_code	IN	VARCHAR2(10)	
p_represented_by_telephone	IN	VARCHAR2(25)	
p_represented_by_extension	IN	VARCHAR2(20)	
p_represented_by_fax_area_code	IN	VARCHAR2(10)	
p_represented_by_fax_number	IN	VARCHAR2(25)	
p_represented_by_time_diff	IN	NUMBER	
p_represented_by_email	IN	VARCHAR2(240)	
p_verify_cp_flag	IN	VARCHAR2(1)	
p_customer_product_id	IN	NUMBER	
p_cp_ref_number	IN	NUMBER	
p_inventory_item_id	IN	NUMBER	
p_inventory_org_id	IN	NUMBER	
p_inventory_item_conc_segs	IN	VARCHAR2	
p_inventory_item_segment1–20	IN	VARCHAR2(40)	
p_inventory_item_vals_or_ids	IN	VARCHAR2(1)	
p_current_serial_number	IN	VARCHAR2(30)	
p_original_order_number	IN	NUMBER	
p_purchase_order_num	IN	VARCHAR2(50)	

Table 8 – 3 Update ServiceRequest Parameters (Page 2 of 4)

Parameter	Usage	Type	Required?
p_problem_description	IN	VARCHAR2(2000)	
p_problem_code	IN	VARCHAR2(30)	
p_exp_resolution_date	IN	DATE	
p_make_public_problem	IN	VARCHAR2(1)	
p_install_location	IN	VARCHAR2(40)	
p_install_customer	IN	VARCHAR2(50)	
p_install_address_line_1	IN	VARCHAR2(240)	
p_install_address_line_2	IN	VARCHAR2(240)	
p_install_address_line_3	IN	VARCHAR2(240)	
p_rma_flag	IN	VARCHAR2(1)	
p_rma_header_id	IN	NUMBER	
p_rma_number	IN	NUMBER	
p_order_type_id	IN	NUMBER	
p_request_segment1–15	IN	VARCHAR2(150)	
p_request_context	IN	VARCHAR2(30)	
p_bill_to_site_use_id	IN	NUMBER	
p_bill_to_contact_id	IN	NUMBER	
p_bill_to_location	IN	VARCHAR2(40)	
p_bill_to_customer	IN	VARCHAR2(50)	
p_bill_address_line_1	IN	VARCHAR2(240)	
p_bill_address_line_2	IN	VARCHAR2(240)	
p_bill_address_line_3	IN	VARCHAR2(240)	
p_bill_to_contact	IN	VARCHAR2(100)	
p_ship_to_site_use_id	IN	NUMBER	
p_ship_to_contact_id	IN	NUMBER	
p_ship_to_location	IN	VARCHAR2(40)	
p_ship_to_customer	IN	VARCHAR2(50)	

Table 8 – 3 Update ServiceRequest Parameters (Page 3 of 4)

Parameter	Usage	Type	Required?
p_ship_address_line_1	IN	VARCHAR2(240)	
p_ship_address_line_2	IN	VARCHAR2(240)	
p_ship_address_line_3	IN	VARCHAR2(240)	
p_ship_to_contact	IN	VARCHAR2(100)	
p_problem_resolution	IN	VARCHAR2(2000)	
p_resolution_code	IN	VARCHAR2(30)	
p_act_resolution_date	IN	DATE	
p_make_public_resolution	IN	VARCHAR2(1)	
p_audit_comments	IN	VARCHAR2(2000)	
p_called_by_workflow	IN	VARCHAR2(1)	
p_workflow_process_id	IN	NUMBER	
p_web_entry_flag	IN	VARCHAR2(1)	
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	
p_return_status	OUT	VARCHAR2(1)	
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_call_id	OUT	NUMBER	

Table 8 – 3 Update ServiceRequest Parameters (Page 4 of 4)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_id Conditionally Required; Not Null **NUMBER**

Enter the service request's system–generated ID. If you do not enter a request ID, you must enter a request number.

p_request_number Conditionally Required; Not Null **VARCHAR2(64)**

Enter the user–visible service request number. If you do not enter a number, you must enter a request ID.

p_type_id Not Null **NUMBER**

Enter a valid and active service request type ID. If the service request has an active workflow associated with it, you cannot update this column in CS_INCIDENTS_ALL. If you enter both a name and an ID, the ID is used and the name ignored.

p_type_name **VARCHAR2(30)**

Enter the service request type name. If you entered a value for *p_type_id*, you need not supply a name here.

p_status_id Not Null **NUMBER**

Enter a valid and active service request status ID. If you enter both a name and an ID, the ID is used and the name ignored. If you update this value to a closed status while an active workflow process is associated with the service request, the interface will abort the workflow.

Default Value 1 (open)

p_status_name **VARCHAR2(30)**

Enter the service request status name. If you entered a value for *p_status_id*, you need not supply a name here. If you update this value to a closed status while an active workflow process is associated with the service request, the interface will abort the workflow.

p_severity_id Not Null **NUMBER**

Enter a valid and active service request severity ID. If you enter both a name and an ID, the ID is used and the name ignored.

p_severity_name **VARCHAR2(30)**

Enter the service request severity name. If you entered a value for *p_severity_id*, you need not supply a name here.

p_urgency_id **NUMBER**

Enter a valid and active service request urgency ID. If you enter both a name and an ID, the ID is used and the name ignored.

p_urgency_name **VARCHAR2(30)**

Enter the service request urgency name. If you entered a value for *p_urgency_id*, you need not supply a name here.

p_closed_date **DATE**

Enter the date this service request was closed. This date must be later than the value of *p_request_date*. If the request's status is not closed, any value entered here is ignored.

Default Value SYSDATE if the new status is a closed status

p_owner_id Not Null **NUMBER**

Identify a valid and active employee ID as the service request's owner. If the service request has an active workflow associated with it, you cannot update this column in CS_INCIDENTS_ALL.

p_publish_flag **VARCHAR2**

Indicate whether the service request is published (entered into your knowledge base), if you have enabled the ConText server option. You can set this flag to TRUE only if *Service: Publish Flag Update Allowed* is set to Yes.

p_summary Not Null **VARCHAR2(80)**

Enter a summary line.

p_verify_request_flag Not Null **VARCHAR2(1)**

Indicate whether the service request is entered in verified mode. If this flag is TRUE, the interface validates the following attributes: *customer_id*, *contact_id*, *bill_to_site_use_id*, *bill_to_contact_id*, *ship_to_site_use_id*, and *ship_to_contact_id*. You can enter FALSE only if CS_INCIDENTS_ALL.RECORD_IS_VALID_FLAG was not previously set.

p_customer_id **NUMBER**

Enter the customer ID associated with the service request. This parameter is used only if *p_verify_request_flag* is set.

p_contact_id **NUMBER**

Enter the customer contact ID associated with the service request. This parameter is ignored if you do not set *p_verify_request_flag*.

p_contact_time_diff **NUMBER**

Enter the time zone difference between your location and that of the customer contact. The difference must be between –24 and 24.

p_represented_by_time_diff **NUMBER**

Enter the time zone difference between your location and that of the customer representative. The difference must be between –24 and 24.

p_verify_cp_flag Not Null **VARCHAR2(1)**

Indicate whether to validate customer product IDs against your installed base. You can enter TRUE if and only if either *p_verify_request_flag* is set or CS_INCIDENTS_ALL.RECORD_IS_VALID_FLAG was previously set.

p_customer_product_id Not Null **NUMBER**

Enter the unique ID of a customer product in your installed base.

p_inventory_org_id **NUMBER**

Specify the inventory organization ID for the organization where the item is located. This is part of the unique key that identifies your inventory item and corresponds to the ORGANIZATION_ID column in the MTL_SYSTEM_ITEMS table. You must enter a value here to convert inventory item IDs.

Default Value *OE: Item Validation Organization*

p_inventory_item_vals_or_ids **VARCHAR2(1)**

Indicate whether the input segments are values ('V') or hidden IDs ('I'). If you input values, the interface expects one value for each displayed

segment. If you use IDs, the interface expects one ID for each enabled segment, regardless of whether it is displayed.

Default Value 'V'

p_exp_resolution_date **DATE**

Enter the service request's expected resolution date. This date must be after the date passed by *p_request_date*.

p_rma_flag Not Null **VARCHAR2(1)**

Indicate whether an RMA (return material authorization) is assigned to the service request. If this parameter is TRUE, the interface performs validation on *p_rma_header_id*. You can enter TRUE if and only if either *p_verify_request_flag* is set or CS_INCIDENTS_ALL.RECORD_IS_ VALID_FLAG was previously set.

p_order_type_id **NUMBER**

Enter the ID of the RMA's order type. You must enter a value here to convert the header ID.

p_act_resolution_date **DATE**

Enter the service request's actual resolution date. This date must be after the date passed by *p_request_date*.

p_audit_comments **VARCHAR2(2000)**

Enter comments for the audit record.

p_called_by_workflow **VARCHAR2(1)**

This parameter is for internal use only.

p_workflow_process_id **NUMBER**

This parameter is for internal use only.

p_web_entry_flag **VARCHAR2(1)**

Indicate whether the service request update is entered from Oracle Self–Service Web Applications.

Default Value FND_API.G_FALSE

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

Update_Status

The Update_Status interface updates the status of service requests stored in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 30.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required
p_request_id	IN	NUMBER	
p_request_number	IN	VARCHAR2(64)	
p_status_id	IN	NUMBER	
p_status	IN	VARCHAR2(30)	
p_closed_date	IN	DATE	
p_audit_comments	IN	VARCHAR2(2000)	
p_called_by_workflow	IN	VARCHAR2(1)	
p_workflow_process_id	IN	NUMBER	
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	
p_return_status	OUT	VARCHAR2(1)	

Table 8 – 4 Update_Status Parameters (Page 1 of 2)

Parameter	Usage	Type	Required?
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_call_id	OUT	NUMBER	

Table 8 – 4 Update_Status Parameters (Page 2 of 2)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_id Conditionally Required; Not Null **NUMBER**

Enter the service request's system–generated ID. If you do not enter a request ID, you must enter a request number.

p_request_number Conditionally Required; Not Null **VARCHAR2(64)**

Enter the user–visible service request number. If you do not enter a request number, you must enter a request ID.

p_status_id Conditionally Required; Not Null **NUMBER**

Enter a valid and active service request status ID. If you do not enter a status ID, you must enter a status name. If you update this value to a closed status while an active workflow process is associated with the service request, the interface will abort the workflow.

p_status Conditionally Required; Not Null **VARCHAR2(30)**

Enter the service request status name. If you do not enter a status name, you must enter a status ID. If you update this value to a closed status while an active workflow process is associated with the service request, the interface will abort the workflow.

p_closed_date **DATE**

Enter the date this service request was closed. This date must be later than the value of *p_request_date*. If the request's status is not closed, any value entered here is ignored.

Default Value SYSDATE if the new status is a closed status

p_called_by_workflow **VARCHAR2(1)**

This parameter is for internal use only.

p_workflow_process_id **NUMBER**

This parameter is for internal use only.

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

Update_Severity

The Update_Severity interface updates the severity of service requests stored in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 33.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required

Table 8 – 5 Update_Severity Parameters (Page 1 of 2)

Parameter	Usage	Type	Required?
p_request_id	IN	NUMBER	
p_request_number	IN	VARCHAR2(64)	
p_severity_id	IN	NUMBER	
p_severity	IN	VARCHAR2(30)	
p_audit_comments	IN	VARCHAR2(2000)	
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	
p_return_status	OUT	VARCHAR2(1)	
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_call_id	OUT	NUMBER	

Table 8 – 5 Update_Severity Parameters (Page 2 of 2)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_id Conditionally Required; Not Null **NUMBER**

Enter the service request's system–generated ID. If you do not enter a request ID, you must enter a request number.

p_request_number Conditionally Required; Not Null **VARCHAR2(64)**

Enter the user–visible service request number. If you do not enter a request number, you must enter a request ID.

p_severity_id Conditionally Required; Not Null **NUMBER**

Enter a valid and active service request severity ID. If you do not enter a severity ID, you must enter a severity name.

p_severity Conditionally Required; Not Null **VARCHAR2(30)**

Enter the service request severity name. If you do not enter a severity name, you must enter a severity ID.

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

Update_Urgency

The Update_Urgency interface updates the urgency of service requests stored in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 36.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required
p_request_id	IN	NUMBER	
p_request_number	IN	VARCHAR2(64)	
p_urgency_id	IN	NUMBER	
p_urgency	IN	VARCHAR2(30)	
p_audit_comments	IN	VARCHAR2(2000)	
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	

Table 8 – 6 Update_Urgency Parameters (Page 1 of 2)

Parameter	Usage	Type	Required?
p_return_status	OUT	VARCHAR2(1)	
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_call_id	OUT	NUMBER	

Table 8 – 6 Update_Urgency Parameters (Page 2 of 2)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_id Conditionally Required; Not Null **NUMBER**

Enter the service request's system–generated ID. If you do not enter a request ID, you must enter a request number.

p_request_number Conditionally Required; Not Null **VARCHAR2(64)**

Enter the user–visible service request number. If you do not enter a request number, you must enter a request ID.

p_urgency_id Conditionally Required **NUMBER**

Enter a valid and active service request urgency ID. If you do not enter a urgency ID, you must enter a urgency name.

p_urgency Conditionally Required **VARCHAR2(30)**

Enter the service request urgency name. If you do not enter a urgency name, you must enter a urgency ID.

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

Update_Owner

The Update_Owner interface updates the owner of service requests stored in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 39.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required
p_request_id	IN	NUMBER	
p_request_number	IN	VARCHAR2(64)	
p_owner_id	IN	NUMBER	Required
p_audit_comments	IN	VARCHAR2(2000)	
p_called_by_workflow	IN	VARCHAR2(1)	
p_workflow_process_id	IN	NUMBER	
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	
p_return_status	OUT	VARCHAR2(1)	
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_call_id	OUT	NUMBER	

Table 8 – 7 Update_Owner Parameters (Page 1 of 1)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_id Conditionally Required; Not Null **NUMBER**

Enter the service request's system–generated ID. If you do not enter a request ID, you must enter a request number.

p_request_number Conditionally Required; Not Null **VARCHAR2(64)**

Enter the user–visible service request number. If you do not enter a request number, you must enter a request ID.

p_owner_id Not Null **NUMBER**

Identify a valid and active employee ID as the service request's owner. If the service request has an active workflow associated with it, you cannot update this column in CS_INCIDENTS_ALL.

p_called_by_workflow **VARCHAR2(1)**

This parameter is for internal use only.

p_workflow_process_id **NUMBER**

This parameter is for internal use only.

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

Update_Problem_Code

The Update_Problem_Code interface updates the problem code of service requests stored in the CS_INCIDENTS_ALL table. The table below lists all inbound and outbound parameters. Each parameter is optional unless otherwise specified. Additional information for some parameters is provided below starting on page 8 – 41.

Parameter	Usage	Type	Required?
p_api_version	IN	NUMBER	Required
p_init_msg_list	IN	VARCHAR2(1)	
p_commit	IN	VARCHAR2(1)	
p_resp_appl_id	IN	NUMBER	
p_resp_id	IN	NUMBER	
p_user_id	IN	NUMBER	
p_login_id	IN	NUMBER	
p_org_id	IN	NUMBER	Conditionally required
p_request_id	IN	NUMBER	
p_request_number	IN	VARCHAR2(64)	
p_problem_code	IN	VARCHAR2(30)	Required
p_comments	IN	VARCHAR2(2000)	
p_public_comment_flag	IN	VARCHAR2(1)	
p_return_status	OUT	VARCHAR2(1)	
p_msg_count	OUT	NUMBER	
p_msg_data	OUT	VARCHAR2(2000)	
p_call_id	OUT	NUMBER	

Table 8 – 8 Update_Problem_Code Parameters (Page 1 of 1)

p_api_version Required **NUMBER**

This parameter is used to compare the incoming interface call's version number with the current version number. An error is returned if the version numbers are incompatible.

p_init_msg_list **VARCHAR2(1)**

Optionally request that the interface initialize the message list on your behalf. This reduces the number of calls your script must make to execute the interface.

Default Value FND_API.G_FALSE

p_commit **VARCHAR2(1)**

Optionally request that the interface commit data for you after it completes its function.

Default Value FND_API.G_FALSE

p_resp_appl_id **NUMBER**

Enter the application ID. The application determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_APPL_ID

p_resp_id **NUMBER**

Enter the responsibility ID. The responsibility determines which profile values are used as default.

Default Value FND_GLOBAL.RESP_ID

p_user_id **NUMBER**

Enter a valid Oracle Applications user ID.

Default Value FND_GLOBAL.USER_ID

p_login_id **NUMBER**

Enter a valid login session ID.

Default Value FND_GLOBAL.LOGIN_ID

p_org_id Conditionally Required **NUMBER**

Identify the operating unit ID for the operating unit . If Multi–Org is enabled, you must enter a value here. If you do not use Multi–Org, any value you enter for this parameter is ignored.

Default Value CLIENT_INFO or *MO: Operating Unit*

p_request_id Conditionally Required; Not Null **NUMBER**

Enter the service request's system–generated ID. If you do not enter a request ID, you must enter a request number.

p_request_number Conditionally Required; Not Null **VARCHAR2(64)**

Enter the user–visible service request number. If you do not enter a request number, you must enter a request ID.

p_problem_code **VARCHAR2(30)**

Enter the problem code associated with the service request.

p_public_comment_flag **VARCHAR2(1)**

Indicate whether the comments are public.

Default Value FND_API.G_FALSE

Running the Interfaces

If you do not want to update a particular column in CS_INCIDENTS_ALL, 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). You cannot update a service request's publish flag if the *Service: Publish Flag Update Allowed* profile is not set.

The following sections include examples for calling some of the interfaces.

Create_ServiceRequest

```
-- Open an internal service request
DECLARE
  return_status        VARCHAR2(1);
  msg_count            NUMBER;
  msg_data             VARCHAR2(2000);
  request_id           NUMBER;
  request_number       VARCHAR2(64);
  call_id              NUMBER;
  itemkey              VARCHAR2(240);
  return_status_wkflw  VARCHAR2(1);
  message              VARCHAR2(2000);
BEGIN
    -- have the API initialize message list
    -- have the API commit its work
```

```
  CS_ServiceRequest_PUB.Create_ServiceRequest (
    p_api_version         => 1.0,
    p_init_msg_list       => FND_API.G_TRUE,
    p_commit              => FND_API.G_TRUE,
    p_return_status       => return_status,
    p_msg_count           => msg_count,
    p_msg_data            => msg_data,
    p_type_name           => 'Internal',
    p_status_name         => 'Open',
    p_severity_name       => 'High',
    p_urgency_name        => 'Inoperable',
    p_summary             => 'All printers do not respond.',
    p_filed_by_emp_flag   => FND_API.G_TRUE,
    p_employee_number     => '12345',
    p_verify_cp_flag      => FND_API.G_FALSE,
    p_problem_code        => 'PRINT',
    p_rma_flag            => FND_API.G_FALSE,
    p_request_id          => request_id,
    p_request_number      => request_number,
    p_call_id             => call_id,
    p_itemkey             => itemkey,
    p_return_status_wkflw => return_status_wkflw);
  -- Print error messages or warnings
  IF msg_count > 0 THEN
    FOR counter IN REVERSE 1..msg_count LOOP
      message := FND_MSG_PUB.Get(counter, FND_API.G_FALSE);
      DBMS_OUTPUT.Put_Line('  MSG(' || TO_CHAR(counter) || '): ' ||
      message);
    END LOOP;
    FND_MSG_PUB.Delete_Msg;
  END IF;
END;
```

Update_ServiceRequest

```
-- use this API to close a service request and enter the
-- problem resolution
DECLARE
  return_status       VARCHAR2(1);
  msg_count           NUMBER;
  msg_data            VARCHAR2(2000);
  call_id             NUMBER;
  message             VARCHAR2(2000);
BEGIN
  CS_ServiceRequest_PUB.Update_ServiceRequest (
    p_api_version         => 1.0,
    p_return_status       => return_status,
    p_msg_count           => msg_count,
    p_msg_data            => msg_data,
```

```
    p_request_number      => '123456',
    p_status_name         => 'Closed',
    p_problem_resolution  => 'Replaced lens.',
    p_call_id             => call_id);
  -- check return status
  IF return_status = FND_API.G_RET_STS_SUCCESS THEN
    DBMS_OUTPUT.Put_Line('Result: Successful');
  ELSIF return_status = FND_API.G_RET_STS_ERROR THEN
    DBMS_OUTPUT.Put_Line('Result: Error');
  ELSIF return_status = FND_API.G_RET_STS_UNEXP_ERROR THEN
    DBMS_OUTPUT.Put_Line('Result: Unexplained Error');
  END IF;
END;
```

Update_Status

```
-- use this API to close several service requests at once
DECLARE
  return_status       VARCHAR2(1);
  msg_count           NUMBER;
  msg_data            VARCHAR2(2000);
  call_id             NUMBER;
  message             VARCHAR2(2000);
BEGIN
  FOR req_num IN 1..10 LOOP
    -- don't have API commit for each transaction
    CS_ServiceRequest_PUB.Update_Status (
      p_api_version         => 1.0,
      p_commit              => FND_API.G_FALSE,
      p_return_status       => return_status,
      p_msg_count           => msg_count,
      p_msg_data            => msg_data,
      p_request_id          => req_num,
      p_status              => 'Closed',
      p_call_id             => call_id);
    IF return_status != FND_API.G_RET_STS_SUCCESS THEN
      DBMS_OUTPUT.Put_Line('Cannot close request #' || req_num);
    END IF;
  END LOOP;
  -- commit work
  COMMIT;
END;
```

Update_Owner

```
-- reassign several service requests
DECLARE
  return_status       VARCHAR2(1);
```

```
  msg_count           NUMBER;
  msg_data            VARCHAR2(2000);
  call_id             NUMBER;
  message             VARCHAR2(2000);
BEGIN
  FOR req_num IN 100..199 LOOP
    CS_ServiceRequest_PUB.Update_Owner (
      p_api_version         => 1.0,
      p_commit              => FND_API.G_FALSE,
      p_return_status       => return_status,
      p_msg_count           => msg_count,
      p_msg_data            => msg_data,
      p_request_id          => req_num,
      p_owner_id            => 1002348,
      p_call_id             => call_id);
    IF return_status != FND_API.G_RET_STS_SUCCESS THEN
      DBMS_OUTPUT.Put_Line('Cannot reassign request #' || req_num);
    END IF;
  END LOOP;
  COMMIT;
END;
```

CHAPTER

9

Oracle Work in Process Open Interfaces

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

Open Job and Schedule Interface

Using the Open Job and Schedule Interface you can import planned orders, planned order update recommendations, and suggested repetitive schedules en mass.

You can insert records into the Job and Schedule Interface (WIP_JOB_ SCHEDULE_ INTERFACE) table from any source including planning systems, order entry systems, finite scheduling packages, production line sequencing programs, spreadsheets, and even custom entry forms. For example, if 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.

The Import Jobs and Schedules window is used to launch the WIP Mass Load program. The WIP Mass Load program validates records in the Job and Schedule Interface table and implements imported records as new discrete jobs, updated discrete jobs, and pending repetitive schedules accordingly.

See Also

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

Functional Overview

Figure 9–1 depicts the three types of order recommendation inputs that the Job and Schedule Interface supports as well as their corresponding Oracle Work in Process discrete job or repetitive schedule output.

Figure 9 – 1

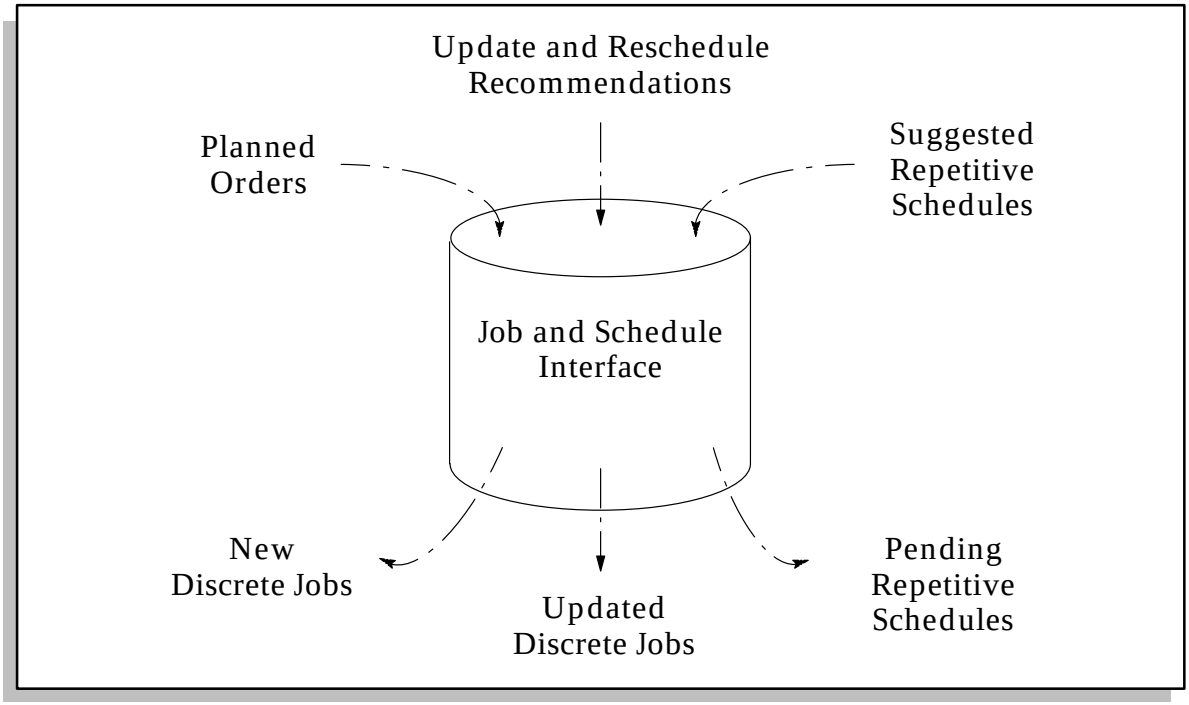
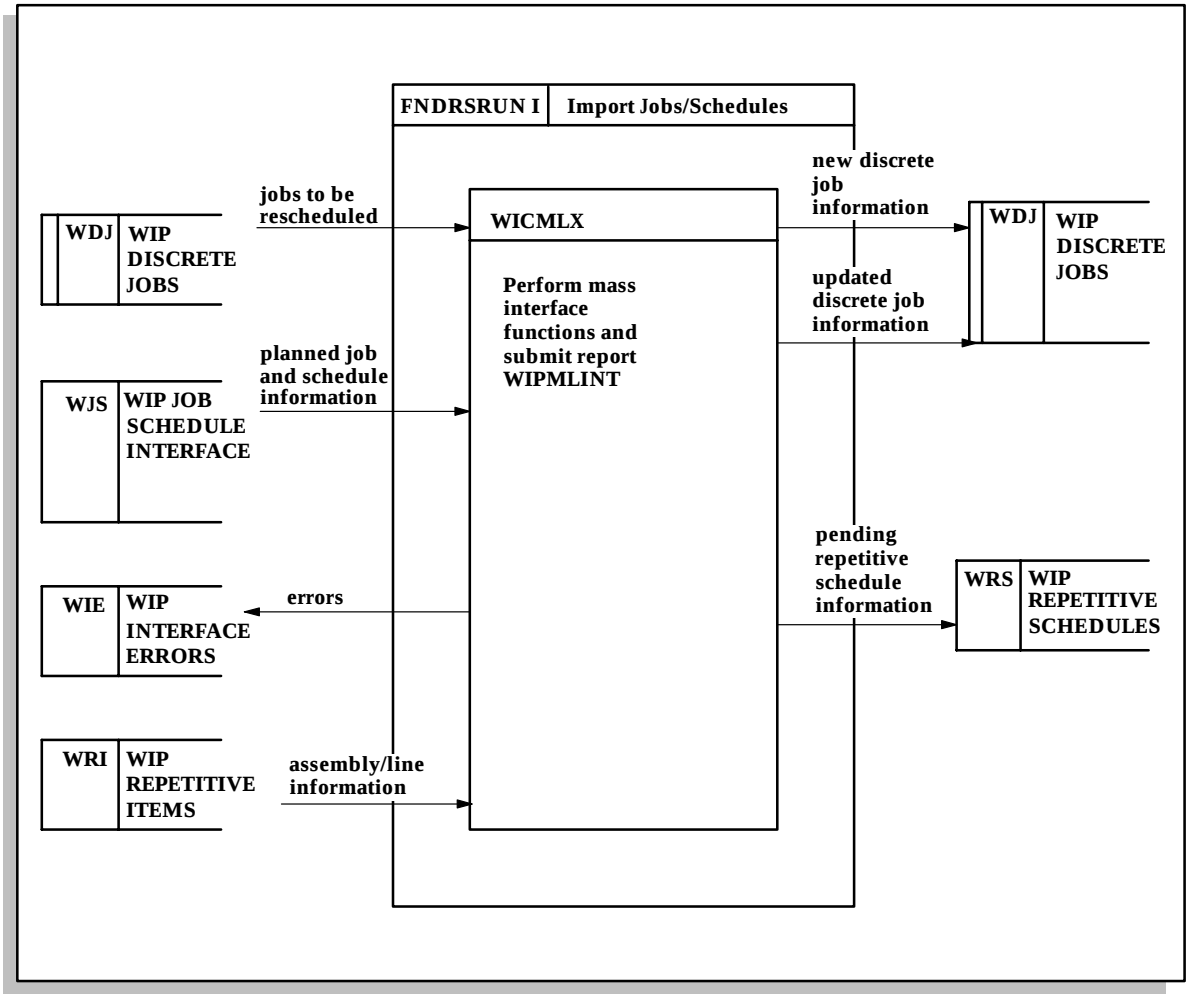


Figure 9–2 shows the key tables and programs that comprise the Job and Schedule Interface.

Figure 9 – 2



You must write a PL/SQL program or use another program to insert a single record (row) into the WIP_JOB_SCHEDULE_INTERFACE table for each planned order, update recommendation, or suggested repetitive schedule. You can then use the Import Jobs and Schedules window to launch the WIP Mass Load (WICMLX) concurrent program for any group of records as identified by their GROUP_ID.

The WIP Mass Load program does all of the following:

- Validates the data in the WIP_JOB_SCHEDULE_INTERFACE table
- Derives values for additional columns
- Creates new discrete jobs, updates existing jobs, and / or creates pending repetitive schedules
- Optionally launches the Job and Schedule Interface Report (WIPMLINT)
- Deletes successfully processed records from the interface table

The Job and Schedule Interface Report lists both successfully processed and failed records. Failed records can be resubmitted using the Pending Jobs and Schedules window.

See Also

Job and Schedule 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 Job and Schedule Interface

The Job and Schedule Interface requires no additional setup steps beyond those already required to set up discrete and repetitive manufacturing. All processing is initiated via the Import Jobs and Schedules window and managed by the concurrent manager. As such, the concurrent manager 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 Job and Schedule Interface

You must write a PL/SQL program or use another program to insert your planned orders, update recommendations, and suggested repetitive schedule records from your source system into the WIP_JOB_SCHEDULE_INTERFACE table.

WIP_JOB_SCHEDULE_INTERFACE Table Description

Table 9–1 describes the WIP_JOB_SCHEDULE_ INTERFACE:

Legend	
Type Number	**Load/Update Type Description**
1	Create Standard Discrete Job
2	Create Pending Repetitive Schedule
3	Update Standard or Non–Standard Discrete Job
4	Non–Standard Discrete Job

Note: The numbers under Required, Optional/Derived if Null, Optional, and Derived or Ignored in Table 9–1 indicate that the referenced column is used to do the following:

[WIP_JOB_SCHEDULE_INTERFACE] Column Name	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored
LOAD_TYPE	Number	1,2,3,4			
PROCESS_PHASE	Number	1,2,3,4			
PROCESS_STATUS	Number	1,2,3,4			
GROUP_ID	Number	1,2,3,4			
INTERFACE_ID	Number				1,2,3,4
LAST_UPDATE_DATE	Date	1,2,3,4			
LAST_UPDATED_BY	Number	1,2,3,4			
LAST_UPDATED_BY_NAME	Varchar2(100)				
CREATION_DATE	Date	1,2,3,4			

Table 9 – 1 WIP Job and Schedule Interface (Page 1 of 3)

[WIP_JOB_SCHEDULE_INTERFACE] Column Name	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored
CREATED_BY_NAME	Varchar2(100)	1,2,3,4			
CREATED_BY	Number				
LAST_UPDATE_LOGIN	Number			1,2,3,4	
REQUEST_ID	Number				1,2,3,4
PROGRAM_ID	Number				1,2,3,4
PROGRAM_APPLICATION_ID	Number				1,2,3,4
PROGRAM_UPDATE_DATE	Date				1,2,3,4
SOURCE_CODE	Varchar2(30)			1,2,3,4	
SOURCE_LINE_ID	Number			1,2,3,4	
STATUS_TYPE	Number		1,4	3	2
FIRST_UNIT_START_DATE	Date	1,2,4		3	
FIRST_UNIT_COMPLETION_DATE					
LAST_UNIT_START_DATE					
LAST_UNIT_COMPLETION_DATE					
SCHEDULING_METHOD	Number		1,3,4		2
PROCESSING_WORK_DAYS	Number	2			1,3,4
DAILY_PRODUCTION_RATE	Number	2			1,3,4
LINE_ID	Number	2		1,3,4	
LINE_CODE	Varchar2(10)				
PRIMARY_ITEM_ID	Number	1, 2		4	3
BOM_REFERENCE_ID	Number			4	1,2,3
ROUTING_REFERENCE_ID	Number			4	1,2,3
ROUTING_REVISION	Number		1,2,4		3
ROUTING_REVISION_DATE	Date				
BOM_REVISION	Number		1,2,4		3
BOM_REVISION_DATE	Date				
COMPLETION_SUBINVENTORY	Varchar2(10)		1,4		2,3

Table 9 – 1 WIP Job and Schedule Interface (Page 2 of 3)

[WIP_JOB_SCHEDULE_INTERFACE] Column Name	Type	Required	Optional/ Derived if Null	Optional	Derived or Ignored
COMPLETION_LOCATOR_ID	Number		1,4		2,3
COMPLETION_LOCATOR_SEGMENTS	Varchar2(10)				
WIP_SUPPLY_TYPE	Number		1, 4		2,3
CLASS_CODE	Varchar2(10)	4	1		2,3
LOT_NUMBER	Varchar2(30)		1,4	3	2
JOB_NAME	Varchar2(240)		1,4		2,3
DESCRIPTION	Varchar2(240)		1,2,4	3	
FIRM_PLANNED_FLAG	Number		1,2,4	3	
ALTERNATE_ROUTING_DESIGNATOR	Varchar2(10)			1,4	2,3
ALTERNATE_BOM_DESIGNATOR	Varchar2(10)			1,4	2,3
SCHEDULE_GROUP_ID	Number			1,3,4	2
SCHEDULE_GROUP_NAME	Varchar2(240)				
BUILD_SEQUENCE	Number			1,3,4	2
PROJECT_ID	Number		3 (See Note)	1, 4	2
PROJECT_NUMBER	Varchar2(25)				
TASK_ID	Number		3 (See Note)	1, 4	2
TASK_NUMBER	Varchar2(25)				
DEMAND_CLASS	Varchar2(30)			1,2,4	3
NET_QUANTITY	Number			1,3,4	2
START_QUANTITY	Number			1,3,4	2
WIP_ENTITY_ID	Number	3			1,2,4
REPETITIVE_SCHEDULE_ID	Number				1,2,3,4
ATTRIBUTE_CATEGORY	Varchar2(30)			1,2,3,4	
ATTRIBUTE1 – ATTRIBUTE15	Varchar2(150)			1,2,3,4	

Table 9 – 1 WIP Job and Schedule Interface (Page 3 of 3)

Control Columns

The columns LOAD_TYPE, PROCESS_PHASE, PROCESS_STATUS, and GROUP_ID are control columns for the WIP Mass Load program. Other columns represent the actual data that is inserted into or modified in the WIP_DISCRETE_JOBS and WIP_REPETITIVE_ SCHEDULES tables when records are successfully imported from the Job and Schedule 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, R11.

LOAD_TYPE: The LOAD_TYPE determines whether the current interface record is a planned order, update recommendation, or suggested repetitive schedule. Possible values are as follows:

- 1 Create Standard Discrete Job
- 2 Create Pending Repetitive Schedule
- 3 Update Standard or Non–Standard Discrete Job
- 4 Create Non–Standard Discrete Job

The LOAD_TYPE also controls whether interface table columns are Required, Optional, Optional / Derived if Null, or Derived or Ignored.

PROCESS_PHASE and PROCESS_STATUS: These columns indicate the current status of each record.

Possible PROCESS_PHASE values include:

- 2 Validation
- 3 Explosion
- 4 Complete
- 5 Creation

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 in any stage of 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). You can set the PROCESS_STATUS of status 5 (Warning) records can be set to 1 (Pending) and set the PROCESS_PHASE to 2 (Validation) then resubmit them.

GROUP_ID: The GROUP_ID column is used to group rows in the interface table. When you launch the WIP Mass Load program you specify a GROUP_ID. Only records with this GROUP_ID are selected for processing. A sequence, WIP_JOB_SCHEDULE_INTERFACE_S, SHOULD be used to generate a new, unique GROUP_ID for each batch of rows that you insert into the WIP_JOB_SCHEDULE_INTERFACE table.

Required Columns

You must specify values for columns in this category. Note that in some cases, whether a field is required may depend on the LOAD_TYPE of the record. 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.

Attention: For some required columns — LINE_CODE / LINE_ID, CREATED_BY_NAME / CREATED_BY_ID, LAST_UPDATED_BY / LAST_UPDATED_BY_ID, ORGANIZATION_CODE / ORGANIZATION_ID — as well as some optional columns — SCHEDULE_GROUP_NAME / SCHEDULE_GROUP_ID PROJECT_ID / PROJECT_NUMBER, and TASK_ID / TASK_NUMBER — you can use either the name or the underlying ID. For example, to specify a production line you can use either the LINE_CODE or the LINE_ID. If you specify values for both the name and the underlying ID, the value for the ID (e.g. LINE_ID) is used and the value for the name (e.g. LINE_CODE) is ignored.

Optional/Derived if Null Columns

You can optionally specify values for columns in this category. If you specify a value, the WIP Mass Load program uses it. If you leave it blank (null), an internal default value is used 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 do not specify

a value, the value defaults to 1 (Unreleased). If you do specify a value, that value is used when the job is created. In general, the 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.*

Note: The PROJECT_ID / PROJECT_NUMBER and TASK_ID / TASK_NUMBER columns are interdependent. When loading records that update existing jobs (Load / Update Type #3), the following rules are applied:

Task = Null
Project = Null
If the job has a project and task reference, the values in these field are not overwritten. In other words, they 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 just a project will fail to load.

Project = Null
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.

Additional Information: Completion locators for standard project jobs (Load / Update Type #3) are automatically recreated when a project or task changes.

Optional Columns

You do not have to enter values for columns in this category. However, unlike Optional / Derived if Null columns, 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 you enter in these columns.

Derived or Ignored Columns

You should leave all columns in this category blank (null). These columns are for internal processing only. Values entered in these columns are ignored or overwritten.

Validating Job and Schedule Interface Records

The WIP Mass Load program validates all required and optional data. If the required or optional data you enter is invalid or if required data is missing, the program updates the PROCESS_STATUS for the record to 3 (Error). Specific information about the cause of the failure is written to the WIP_INTERFACE_ERRORS table. You can view this information using the Pending Jobs and Schedules window or by printing the Job and Schedule Interface Report.

If an error that is not specific to the particular record occurs during processing, then the WIP Mass Load program sets the PROCESS_STATUS to 3 (Error) for all remaining records within the chosen GROUP_ID.

The WIP Mass Load program sets the PROCESS_PHASE to 4 (Complete) for rows that pass validation and are successfully processed.

If you choose to print the Job and Schedule 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 Job and Interface Interface Status Report is submitted for printing. 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.

See Also

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

Job and Schedule Interface Report, *Oracle Work in Process User's Guide*

Open Move Transaction Interface

You can use external data collection devices such as bar code readers, automated test equipment, cell controllers, and other manufacturing execution systems to collect move transactions or combined move and completion/return transactions.

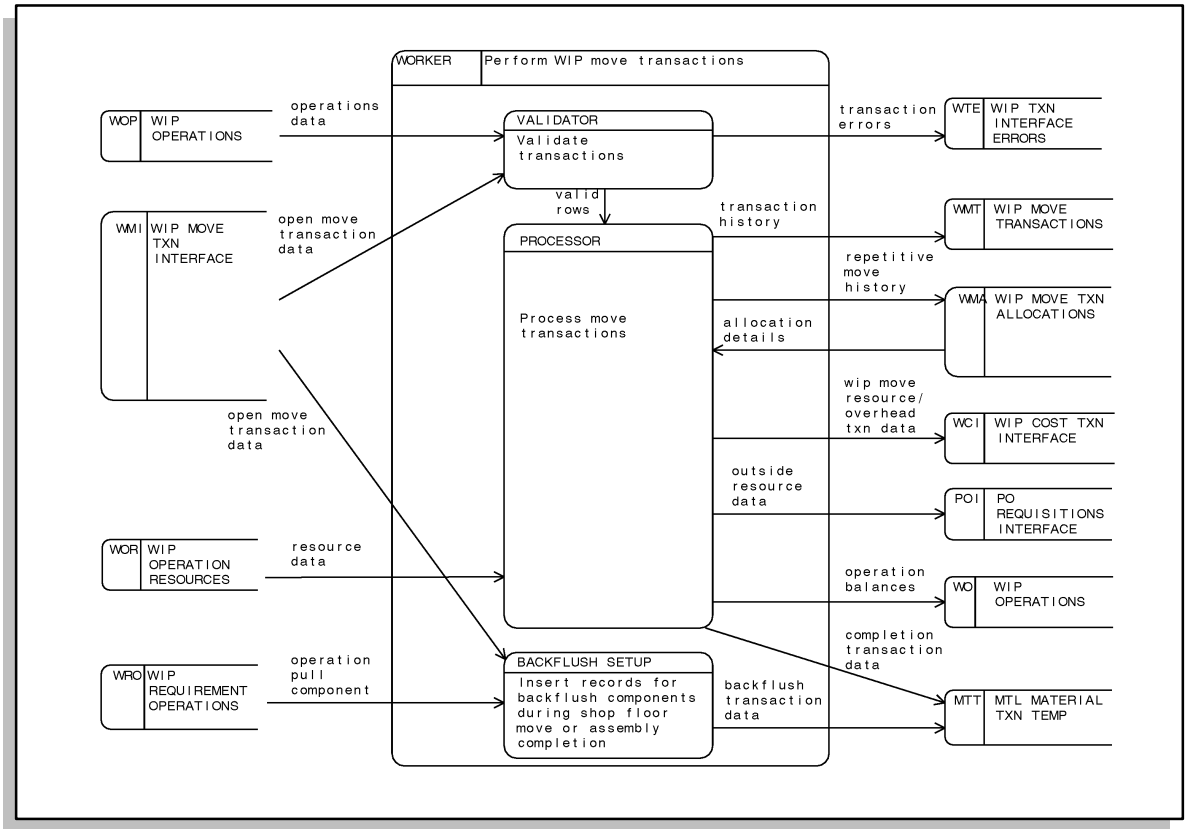
You can load these transactions into the Open Move Transaction Interface for Oracle Work in Process to process. All transactions are validated and invalid transactions are marked so that you can correct and resubmit them.

The purpose of this essay is to explain how to use the Move Transaction Interface so that you can 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 9 – 3



You must write the load program that inserts a single row for each move transaction into the WIP_MOVE_TXN_INTERFACE table. Then the Move Transaction Manager (WICTMS) 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 writing the transaction to history, allocates the move transaction to the correct repetitive schedule (for repetitive manufacturing only), initiates related resource and overhead

transactions, initiates requisitions for outside resources (for outside processing only), updates operations balances, initiates a completion transaction (for combination move and completion/return transactions) and deletes the successfully processed transaction row from the WIP_MOVE_TXN_INTERFACE table.

Finally, the Backflush Setup program determines and initiates related operation pull backflushes.

You can use the Pending Move Transactions window (WIPTSUPD) to review pending move transactions and to update or delete transactions that failed processing due to validation or other errors.

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 you import from external sources.

See Also

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 transaction that can be especially helpful when transacting a large batch of transactions imported through the Move Transaction Interface.

See Also

Transaction Managers, *Oracle Inventory User's Guide*

Inserting Records into the WIP_MOVE_TXN_INTERFACE Table

You must insert your move transactions or combination move and completion transactions into the WIP_MOVE_TXN_INTERFACE table. The system validates each transaction row, derives any additional data as necessary and then processes each transaction.

WIP_MOVE_TXN_INTERFACE Table Description

The following describes the WIP_MOVE_TXN_INTERFACE table:

[WIP_MOVE_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
TRANSACTION_ID	Number		✓	
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number		✓	
LAST_UPDATED_BY_NAME	VarChar2(100)	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number		✓	
CREATED_BY_NAME	VarChar2(100)	✓		
LAST_UPDATE_LOGIN	Number		✓	
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
GROUP_ID	Number		✓	

Table 9 – 2 Move Transaction Interface (Page 1 of 3)

[WIP_MOVE_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
SOURCE_CODE	Varchar2(30)			✓
SOURCE_LINE_ID	Number			✓
PROCESS_PHASE	Number	✓		
PROCESS_STATUS	Number	✓		
ORGANIZATION_ID	Number		✓	✓
ORGANIZATION_CODE	VarChar2(3)	✓		
WIP_ENTITY_ID	Number		✓	✓
WIP_ENTITY_NAME	VarChar2(240)	✓		
ENTITY_TYPE	Number		✓	
PRIMARY_ITEM_ID	Number		✓	
LINE_ID	Number		✓	
LINE_CODE	VarChar2(10)			✓
REPETITIVE_SCHEDULE_ID	Number		✓	
TRANSACTION_DATE	Date	✓		
ACCT_PERIOD_ID	Number		✓	
FM_OPERATION_SEQ_NUM	Number	✓		
FM_OPERATION_CODE	VarChar2(4)		✓	
FM_DEPARTMENT_ID	Number		✓	
FM_DEPARTMENT_CODE	VarChar2(10)		✓	
FM_INTRAOPERATION_STEP_TYPE	Number	✓		
TO_OPERATION_SEQ_NUM	Number	✓		
TO_OPERATION_CODE	VarChar2(4)		✓	
TO_DEPARTMENT_ID	Number		✓	

Table 9 – 2 Move Transaction Interface (Page 2 of 3)

[WIP_MOVE_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
TO_DEPARTMENT_CODE	VarChar2(10)		✓	
TO_INTRAOPERATION_STEP_TYPE	Number	✓		
TRANSACTION_QUANTITY	Number	✓		
TRANSACTION_UOM	VarChar2(3)	✓		
PRIMARY_QUANTITY	Number		✓	
PRIMARY_UOM	VarChar2(3)		✓	
SCRAP_ACCOUNT_ID	Number			✓
REASON_ID	Number		✓	✓
REASON_NAME	VarChar2(30)			✓
REFERENCE	VarChar2(240)			✓
ATTRIBUTE_CATEGORY	Varchar2(30)			✓
ATTRIBUTE1 – ATTRIBUTE15	Varchar2(150)			✓
TRANSACTION_TYPE	Number			✓

Table 9 – 2 Move Transaction Interface (Page 3 of 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.

Required Data

You should set TRANSACTION_TYPE to 1 or null for normal move transactions. If you leave TO_OPERATION_SEQ_NUM and TO_INTRAOPERATION_STEP_TYPE blank then the data is defaulted. The TO_OPERATION_SEQ_NUM is defaulted to the next count point operation in the routing. If there is no count point operation and the TO_OPERATION_SEQ_NUM is blank then the transaction fails validation. The TO_INTRAOPERATION_STEP_TYPE is defaulted to 'Queue'.

You should set TRANSACTION_TYPE to 2 for combination move and completion / return transactions. When TRANSACTION_TYPE is 2, the move transaction processor moves the assemblies to the last operation in the routing and completes the units into or returns them from the completion subinventory / locator. Do not insert data into the TO_OPERATION_SEQ_NUM or TO_INTRAOPERATION_STEP_TYPE for combination move and completion / return transactions.

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)

The column PROCESS_STATUS contains the state of the transaction:

- 1 Pending
- 2 Running
- 3 Error

You should always load 1 (Pending)

The column LINE_CODE is required for repetitive manufacturing transactions only.

Derived Data

The WIP Transaction Validation Engine program derives the columns identified as derived above 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:

- TRANSACTION_ID
- LAST_UPDATED_BY
- CREATED_BY
- LAST_UPDATE_LOGIN
- REQUEST_ID
- PROGRAM_APPLICATION_ID

- PROGRAM_ID
- PROGRAM_UPDATE_DATE
- GROUP_ID

You can optionally insert data into certain derived columns. The WIP Transaction Validation Engine does not override your data. It does, however, validate it. You can insert data into the following derived columns:

- ORGANIZATION_ID
- WIP_ENTITY_ID
- LINE_ID
- REASON_ID

Optional Data

The columns SOURCE_CODE and SOURCE_LINE_ID columns can be used to identify the source of your move transactions. For example, if you collect moves 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.

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 each of the required columns and derive data for each of the derived columns. If data is entered in certain derived columns and optional columns, the WIP Transaction Validation Engine program validates these columns as well.

The system considers the dependencies among the columns in the interface table and only processes columns once the columns 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. ORGANIZATION_ID, in

turn, 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.

However, columns independent of each other can be validated regardless of the status of the other column. For example, WIP_ENTITY_NAME is validated even if REASON_NAME is invalid since WIP_ENTITY_NAME is not dependent on REASON_NAME. Using this behavior, 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 can view both pending and failed move transaction rows in the WIP_MOVE_TXN_INTERFACE table using the Pending Move Transactions window. For failed transactions, you can also view the errors associated to each transaction 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 can update the 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 can check 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. If you choose Select All for Resubmit from the *Special Menu*, the system checks the Resubmit check box of all queried rows. When you save your work, all of these rows become eligible for revalidation and processing.

You can use the Pending Move Transactions window to delete problem rows from the WIP_MOVE_TXN_INTERFACE table. Deletions ensure that you do not have duplicate data when you reload the corrected data from the source application.

Transactions that fail an initial validation can be re–queried then resubmitted 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 Also

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 transactions.

You can load these transactions into the Open Resource Transaction Interface for Oracle Work in Process to process. All transactions are validated and invalid transactions are marked so that you can correct and resubmit them.

The purpose of this essay is to explain how to use the Resource Transaction Interface so that you can 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. Then the Cost Manager (CMCCTM) 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.

Next, the Cost Worker performs the actual resource transaction writing the transaction to history, allocates the resource transaction to the correct repetitive schedule (for repetitive manufacturing only), updates operation resource balances and deletes the successfully processed transaction row from the WIP_COST_TXN_INTERFACE table.

You can use the Pending Resource Transactions window (WIPTSUPD) to review pending transactions and to update or delete transactions that failed processing due to validation or other errors.

Setting Up Resource Transaction Interface

You must perform all 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 you import from external sources.

See Also

Setting Up Resource Management, *Oracle Work in Process User's Guide*

Launching the Cost Manager

You launch the Cost Manager in the Interface Managers window in Oracle Inventory. When you launch the Cost Manager you can specify the resubmit interval and 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 processing of transactions which is especially helpful when transacting a large batch of transactions imported through the Resource Transaction Interface.

See Also

Transaction Managers, *Oracle Inventory User's Guide*

Inserting 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 and then processes each transaction.

WIP_COST_TXN_INTERFACE Table Description

The following describes the WIP_COST_TXN_INTERFACE table:

[WIP_COST_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
TRANSACTION_ID	Number		✓	

Table 9 – 3 WIP Cost Transaction Interface (Page 1 of 4)

[WIP_COST_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
LAST_UPDATE_DATE	Date	✓		
LAST_UPDATED_BY	Number		✓	
LAST_UPDATED_BY_NAME	VarChar2(100)	✓		
CREATION_DATE	Date	✓		
CREATED_BY	Number		✓	
CREATED_BY_NAME	VarChar2(100)	✓		
LAST_UPDATE_LOGIN	Number		✓	
REQUEST_ID	Number		✓	
PROGRAM_APPLICATION_ID	Number		✓	
PROGRAM_ID	Number		✓	
PROGRAM_UPDATE_DATE	Date		✓	
GROUP_ID	Number		✓	
SOURCE_CODE	Varchar2(30)			✓
SOURCE_LINE_ID	Number			✓
PROCESS_PHASE	Number	✓		
PROCESS_STATUS	Number	✓		
TRANSACTION_TYPE	Number	✓		
ORGANIZATION_ID	Number		✓	✓
ORGANIZATION_CODE	VarChar2(3)	✓		
WIP_ENTITY_ID	Number		✓	✓
WIP_ENTITY_NAME	VarChar2(240)	✓		
ENTITY_TYPE	Number		✓	
LINE_ID	Number		✓	

Table 9 – 3 WIP Cost Transaction Interface (Page 2 of 4)

[WIP_COST_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
LINE_CODE	VarChar2(10)			✓
PRIMARY_ITEM_ID	Number		✓	
REPETITIVE_SCHEDULE_ID	Number		✓	
TRANSACTION_DATE	Date	✓		
ACCT_PERIOD_ID	Number		✓	
OPERATION_SEQ_NUM	Number	✓		
RESOURCE_SEQ_NUM	Number	✓		
DEPARTMENT_ID	Number		✓	
DEPARTMENT_CODE	VarChar2(10)		✓	
EMPLOYEE_ID	Number		✓	✓
EMPLOYEE_NUM	VarChar2(30)			✓
RESOURCE_ID	Number		✓	
RESOURCE_CODE	VarChar2(10)		✓	✓
RESOURCE_TYPE	Number		✓	
USAGE_RATE_OR_AMOUNT	Number		✓	
BASIS_TYPE	Number		✓	
AUTOCHARGE_TYPE	Number		✓	
STANDARD_RATE_FLAG	Number		✓	
TRANSACTION_QUANTITY	Number	✓		
TRANSACTION_UOM	VarChar2(3)	✓		
PRIMARY_QUANTITY	Number		✓	
PRIMARY_UOM	VarChar2(3)		✓	
PRIMARY_UOM_CLASS	VarChar2(10)		✓	

Table 9 – 3 WIP Cost Transaction Interface (Page 3 of 4)

[WIP_COST_TXN_INTERFACE] Column Name	Type	Required	Derived	Optional
ACTUAL_RESOURCE_RATE	Number		✓	
CURRENCY_CODE	VarChar2(15)			✓
CURRENCY_CONVERSION_DATE	Date		✓	✓
CURRENCY_CONVERSION_TYPE	VarChar2(10)			✓
CURRENCY_CONVERSION_RATE	Number			✓
CURRENCY_ACTUAL_RESOURCE_RATE	Number			✓
ACTIVITY_ID	Number		✓	✓
ACTIVITY_NAME	VarChar2(10)			✓
REASON_ID	Number		✓	✓
REASON_NAME	VarChar2(30)			✓
REFERENCE	VarChar2(240)			✓
MOVE_TRANSACTION_ID	Number			✓
RCV_TRANSACTION_ID	Number			✓
PO_HEADER_ID	Number			✓
PO_LINE_ID	Number			✓
ATTRIBUTE_CATEGORY	Varchar2(30)			✓
ATTRIBUTE1 – ATTRIBUTE15	Varchar2(150)			✓
DISTRIBUTION_ACCOUNT_ID	Number			✓
RECEIVING_ACCOUNT_ID	Number			✓

Table 9 – 3 WIP Cost Transaction Interface (Page 4 of 4)

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 Data

You should set TRANSACTION_TYPE to 1 for normal resource transactions. You should set TRANSACTION_TYPE to 2 for overhead transactions.

The column PROCESS_PHASE describes the current processing phase of the transaction. The Cost Worker processes each transaction row through the following two phases:

- 1 Resource Validation
- 2 Resource Processing

You should always load 1 (Resource Validation)

The column PROCESS_STATUS contains the state of the transaction:

- 1 Pending
- 2 Running
- 3 Error

You should always load 1 (Pending)

The column RESOURCE_ID column must be left NULL.

The column LINE_CODE is required for repetitive manufacturing transactions only.

Derived Data

The WIP Transaction Validation Engine program derives the columns identified as derived above using foreign key relationships within Oracle Manufacturing.

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:

- TRANSACTION_ID
- LAST_UPDATED_BY
- CREATED_BY
- LAST_UPDATE_LOGIN
- REQUEST_ID

- PROGRAM_APPLICATION_ID
- PROGRAM_ID
- PROGRAM_UPDATE_DATE
- GROUP_ID

You can optionally insert data into certain derived columns. The WIP Transaction Validation Engine does not override your data. It does, however, validate it. You can insert data into the following derived columns:

- ORGANIZATION_ID
- WIP_ENTITY_ID
- LINE_ID
- REASON_ID
- EMPLOYEE_ID
- ACTIVITY_ID

Optional Data

The columns SOURCE_CODE and SOURCE_LINE_ID can be used to identify the source of your resource and overhead transactions. For example, if you collect resources from a bar code reader and a labor data entry form, you could use a different source code to identify each collection method.

The descriptive flexfield attributes in the columns ATTRIBUTE1 through ATTRIBUTE15 map to ATTRIBUTES1 through ATTRIBUTE15 in WIP_TRANSACTIONS.

Costing Option

You can charge non–person resources — resources for which an EMPLOYEE_NUMBER is not specificied — at actual by specifying a rate in the USAGE_RATE_OR_AMOUNT column. If you do not specify an actual usage rate or amount, resource rate or amount is derived from the WIP_OPERATION_RESOURCES table.

Note: Non–person resources cannot be charged at an actual usage rate or amount through the Resource Transactions window. This feature is unique to the Resource Transaction Interface.

You can similarly 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. However for person type resources, if no usage rate or amount is specified and the STANDARD_RATE_FLAG set to No (2), the system derives the usage rate or amount using the employee's hourly labor rate from the WIP_EMPLOYEE_LABOR_RATES table. If an invalid employee ID is specifice, 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 at which to charge your transaction. You can 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 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 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 each of the required columns and derive data for each of the derived columns. If data is entered in certain derived columns and optional columns, the WIP Transaction Validation Engine program validates these columns as well.

The system considers the dependencies among the columns in the interface table and only processes columns once the columns 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 derived. ORGANIZATION_ID, in turn, 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 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 when a column it is dependent upon fails validation. So WIP_ENTITY_NAME is not validated if ORGANIZATION_CODE is invalid.

However, columns independent of each other can be validated regardless of the status of the other column. For example, WIP_ENTITY_NAME is validated even if REASON_NAME is invalid since WIP_ENTITY_NAME is not dependent on REASON_NAME. Using this behavior, 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. For failed transactions, you can also view the errors associated to each transaction by navigating to the Pending Resource Transaction Errors window.

Fixing Failed Rows

You can update the failed resource and overhead transaction rows in the WIP_COST_TXN_INTERFACE table using the Pending Resource Transactions window. Once you have made the necessary changes, you can check 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 *Special Menu*, the system checks the Resubmit check box of all queried rows. When you save your work, all of these rows become eligible for revalidation and processing.

You can use the Pending Resource Transactions form to delete problem rows from the WIP_COST_TXN_INTERFACE table. Deletion ensure

that you do not have duplicate data when you reload the corrected data from the source application.

Transaction that fail an initial validation can be re–queried then resubmitted after correcting 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 and then resubmit the pending resource transaction.

See Also

Processing Pending Resource Transactions, *Oracle Work in Process User's Guide*

WIP Scheduling Interface

The WIP Scheduling Interface consists of an interface table and two stored procedures. It can be used to exchange data between Oracle Work in Process and third party finite scheduling systems. Specifically it can be used to reschedule jobs at either the operation or the operation resource level.

Note: Contact Oracle Consulting for more information about third party products supported through the Oracle Cooperative Applications Initiative.

Functional Overview

Job operation or operation resource information can be loaded into the interface table (WIP_SCHEDULING_INTERFACE), rescheduled, then loaded back into Work in Process. The procedures, LOAD_INTERFACE and LOAD_WIP, help automate the process of loading information into and out of the interface table.

The LOAD_INTERFACE and LOAD_WIP procedures are contained in the WIP_SCHEDULING package. For more information about PL/SQL packages and procedures, see the chapter on Subprograms and the chapter on Packages in the *PL/SQL User's Guide and Reference, Release 8.0.*

Because package architecture is flexible and customizable, the WIP Scheduling Interface can be implemented in a variety of ways. For example, you can write a custom program that calls LOAD_INTERFACE, invokes a third party scheduling program or routine, then calls LOAD_WIP. This process is outlined below:

1. Call LOAD_INTERFACE to insert records from the WIP_OPERATIONS or the WIP_OPERATION_RESOURCES table into the interface table.
2. Invoke your third party finite scheduling program or routine to reschedule the operations or operation resources in the interface table.

   Note: To complete the rescheduling process, interim steps may be required by your third party product. For example, you may need to export data from the WIP Scheduling interface table into a database table in your third party product, reschedule it, then export it back into the WIP Scheduling interface table.

3. Call LOAD_WIP to load the rescheduled information back into the WIP_OPERATIONS or the WIP_OPERATION_RESOURCES table.

   When information is loaded back into Work in Process, job start and completion dates in the WIP_DISCRETE_JOBS table and material requirement dates in the WIP_REQUIREMENT_ OPERATIONS table are updated as required.

The above described implementation process is only an example. Other customized implementations are also possible.

Setting Up the WIP Scheduling Interface

The WIP Scheduling Interface requires no additional setup steps beyond those already required to set up discrete manufacturing. However, implementation specific setups may be required in your third party scheduling product and a custom program that invokes LOAD_WIP and possibly LOAD_INTERFACE, or both, must be written.

Loading into the WIP Scheduling Interface Table

You can use the LOAD_INTERFACE procedure to inserts records into the interface table. The use of LOAD_INTERFACE is optional.

The parameters that are passed to LOAD_INTERFACE determine what information is loaded. The following table lists these parameters. A more detailed explanation of each parameter follows the table.

LOAD_INTERFACE Parameters

Parameter	Datatype	Description
P_WIP_ENTITY_ID	NUMBER	Job
P_ORGANIZATION_ID	NUMBER	Organization
P_JOB_INTERFACE_GROUP_ID	NUMBER	Job and Schedule Interface Group ID
P_GROUP_ID	NUMBER	WIP Scheduling Interface Group ID
P_SCHEDULING_LEVEL	NUMBER	Scheduling Level

Note: You must use both the P_WIP_ENTITY_ID and P_ORGANIZATION_ID parameters or just the P_JOB_INTERFACE_GROUP_ID parameter. Do not use the P_JOB_INTERFACE_GROUP_ID parameter unless you intend to use the WIP Mass Load program to import jobs and schedules at the same time you use LOAD_WIP to load rescheduled operation or operation resource information. See: Import Jobs and Schedules, *Oracle Work in Process User's Guide* and Open Job and Schedule Interface.

P_WIP_ENTITY_ID Parameter

Determines, by job, which operation or operation resource records are loaded into the interface table.

P_ORGANIZATION_ID Parameter

Determines which organization you are loading information from.

P_JOB_INTERFACE_GROUP_ID Parameter

If you specify a value here, LOAD _INTERFACE loads the interface table with operation or operation resource records that are related to jobs in the Open Job and Schedule Interface table (WIP_JOB_SCHEDULE_INTERFACE). For example, if there are ten jobs with a GROUP_ID of 45000 in the WIP_JOB_ SCHEDULE_ INTERFACE table, and you specify 45000 for this parameter, LOAD_INTERFACE loads only operation or operation resource records related to these ten jobs.

P_SCHEDULING_LEVEL Parameter

Determines whether to load job operation or job operation resource records from the WIP_OPERATIONS or WIP_OPERATION_ RESOURCES tables. Valid values are 1, load operation information, or 2, load resource information.

P_GROUP_ID Parameter

Determines how records are grouped. For example, if you have written a custom program to call LOAD_WIP, you could design it so that each time it is executed, this counter is incremented.

WIP_SCHEDULING_INTERFACE Table Description

The following table describes the WIP Scheduling Interface table. As you review this information, please bear in mind that this interface table holds both inbound and outbound information. The information contained in this table pertains to the inbound requirements of the LOAD_WIP procedure. Information about outbound data, loaded into this table by LOAD_INTERFACE, is included in the explanatory text following this table.

Legend	
Scheduling Level	**Description**
1	Operation Dates
2	Operation Resource Dates

[WIP_SCHEDULING_INTERFACE] Column Name	Type	Required	Optional	Derived or Ignored
INTERFACE_ID	Number	1,2		
GROUP_ID	Number	1,2		
WIP_ENTITY_ID	Number	1,2		
ORGANIZATION_ID	Number	1,2		
OPERATION_SEQ_NUM	Number	1		2
RESOURCE_SEQ_NUM	Number	2		1
SCHEDULING_LEVEL	Number	1,2		
OPERATION_START_DATE	Date	1		2
OPERATION_COMPLETION_DATE	Date	1		2
RESOURCE_START_DATE	Date	2		1
RESOURCE_COMPLETION_DATE	Date	2		1
USAGE_RATE_OR_AMOUNT	Number		2	1
PROCESS_PHASE	Number	1,2		

Table 9 – 4 WIP Scheduling Interface (Page 1 of 2)

[WIP_SCHEDULING_INTERFACE] Column Name	Type	Required	Optional	Derived or Ignored
PROCESS_STATUS	Number	1,2		
LAST_UPDATE_DATE	Date	1,2		
LAST_UPDATED_BY	Number	1,2		
LAST_UPDATE_LOGIN	Number		1,2	
CREATION_DATE	Date	1,2		
CREATED_BY	Number	1,2		
REQUEST_ID	Number		1,2	
PROGRAM_APPLICATION_ID	Number		1,2	
PROGRAM_ID	Number		1,2	
PROGRAM_UPDATE_DATE	Date		1,2	

Table 9 – 4 WIP Scheduling Interface (Page 2 of 2)

Control Columns

PROCESS_PHASE and PROCESS_STATUS are control columns.

Valid *PROCESS_PHASE* values include:

- 1 Pending
- 2 Validation

Records loaded into the table by LOAD_INTERFACE program 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.

Valid *PROCESS_STATUS* values include:

- 1 Pending
- 2 Running
- 3 Error

Records loaded into the table by LOAD_INTERFACE are assigned a value of 2. Only records with a process status of 2 are picked up by the LOAD_WIP. If LOAD_WIP fails in any stage of processing a record, the process status of that record and all other records related to that record are set to 3 (Error).

Required Columns

Depending on the scheduling level chosen, some columns may or may not be required. Scheduling level, however, is always required.

Valid *SCHEDULING_LEVEL* values include:

- 1 Operations
- 2 Resources

If you are rescheduling operations, OPERATION_SEQ_NUM, OPERATION_START_DATE and OPERATION_COMPLETION_DATE are required.

If you are rescheduling resources, RESOURCE_SEQ_NUM, RESOURCE_START_DATE and RESOURCE_COMPLETION_DATE are required.

GROUP_ID is required and is used to group (batch) rows in the interface table. Only records with this GROUP_ID are processed by LOAD_WIP.

Optional Columns

You do not have to enter values for columns in this category. For example, the you can optionally specify a USAGE_RAGE_OR_ AMOUNT when rescheduling operation resources.

Rescheduling Operations or Operation Resources

In most implementations, a third party scheduling program reschedules the operation or operation resource information. Exactly how rescheduling is accomplished depends on the specific program being used. However, no matter how rescheduling is done, the process phase of each record must be updated from 1 (Pending) to 2 (Validation). LOAD_WIP only picks up records with a process phase of 2.

Loading into Work in Process

You must use the LOAD_WIP procedure to load rescheduled records back into Work in Process. LOAD_WIP has only one parameter, P_GROUP_ID.

LOAD_WIP Parameter

Parameter	Datatype	Description
P_GROUP_ID	NUMBER	WIP Scheduling Interface Group ID

The P_GROUP_ID parameter determines how records are grouped. You can only load records from one group at a time.

Validating WIP Scheduling Interface Records

LOAD_WIP validates all required and optional data before it loads information back into Work in Process. The information that is required, and thus the validation performed, is based on the scheduling level, either 1 or 2, of the records in the group.

If you are loading operations — scheduling level is 1 — operation start and completion dates are also validated. They cannot be null and the operation completion date must be greater than or equal to the operation start date.

If you are loading operation resources — scheduling level is 2 — resource start and completion dates are validated. They cannot be null and the resource completion date must be greater than or equal to the

resource start date. If a usage rate exists, it is also validated. It must be greater or equal to zero.

LOAD_WIP also validates information based on data in other tables. For example, you can only load operation or operation resource information for jobs that have statuses of 1=Unreleased, 3= Released, 4=Complete, or 6=On Hold.

If a record fails validation, either because required or optional data is invalid or because required data is missing, the process status of the failed record is set to 3 (Error). When a single record within a group fails validation, all records within that group fail validation and their process statuses are set to 3 (Error).

The view, WIP_SCHEDULING_ERRORS_V, can be used to view failed records. Successfully processed rows are deleted from the WIP_SCHEDULING_INTERFACE table. Unsuccessfully processed records must be manually removed from the table using PL/SQL.

See Also

Oracle Work in Process Technical Reference Manual

Index

D

F

I

J

M

O

P

R

S

T

W

Reader's Comment Form

Oracle® Manufacturing, Distribution, Sales and Service Open Interfaces Manual A57332–01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information we use 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? What did you like least about it?

If you find any errors or have any other suggestions for improvement, please indicate the topic, chapter, and page number below:

_____

_____

_____

_____

_____

_____

_____

_____

Please send your comments to:

Oracle Applications Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065 USA
Phone: (650) 506–7000 Fax: (650) 506–7200

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

_____

_____

_____

Thank you for helping us improve our documentation.