ORACLE
JD EDWARDS ENTERPRISEONE

EnterpriseOne Order Promising 8.12.1 Guide

May 2010

ORACLE

EnterpriseOne Order Promising 8.12.1 Guide
SKU e1901sop-b0510

Trademark Notice

License Restrictions Warranty/Consequential Damages Disclaimer

Warranty Disclaimer

Restricted Rights Notice

Hazardous Applications Notice

Third Party Content, Products, and Services Disclaimer

Open Source Disclosure

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle's JD Edwards EnterpriseOne products and the following disclaimers are provided:

Apache Software Foundation

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment:

   "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

   Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.
4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org.
5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see http://www.apache.org/.

Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.

ptmalloc

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the name of Wolfram Gloger may not be used in any advertising or publicity relating to the software.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL WOLFRAM GLOGER BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Sleepycat Software

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
4. This product includes software developed by the University of California, Berkeley and its contributors.
5. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Tool Command Language (TCL)

GOVERNMENT USE: If you are acquiring this software on behalf of the U.S. government, the Government shall have only "Restricted Rights" in the software and related documentation as defined in the Federal Acquisition Regulations (FARs) in Clause 52.227.19 (c) (2). If you are acquiring the software on behalf of the Department of Defense, the software shall be classified as "Commercial Computer Software" and the Government shall have only "Restricted Rights" as defined in Clause 252.227-7013 (c) (1) of DFARs. Notwithstanding the foregoing, the authors grant the U.S. Government and others acting in its behalf permission to use and distribute the software in accordance with the terms specified in this license.

Independent JPEG Group

This product includes software developed by the Independent JPEG Group. Copyright © 1991-1998 The authors make NO WARRANTY or representation, either express or implied, with respect to this software, its quality, accuracy, merchantability, or fitness for a particular purpose. This software is provided "AS IS", and you, its user, assume the entire risk as to its quality and accuracy.

Henry Spencer's Regular Expression Library (REGEX)

This product includes software developed by Henry Spencer. Copyright © 1992, 1993, 1994, 1997 This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California. Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

XBAE

Copyright © 1991, 1992 Bell Communications Research, Inc. (Bellcore)

Copyright © 1995-99 Andrew Lister

All Rights Reserved.

Permission to use, copy, modify and distribute this material for any purpose and without fee is hereby granted, provided that the above copyright notices and this permission notice appear in all copies, and that the name of any author not be used in advertising or publicity pertaining to this material without the specific, prior written permission of an authorized representative of Bellcore and current maintainer.

BELLCORE AND OTHER CONTRIBUTORS MAKE NO REPRESENTATIONS AND EXTEND NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR ANY PARTICULAR PURPOSE, AND THE WARRANTY AGAINST INFRINGEMENT OF PATENTS OR OTHER INTELLECTUAL PROPERTY RIGHTS. THE SOFTWARE IS PROVIDED "AS IS", AND IN NO EVENT SHALL ANY AUTHOR OR ANY OF THEIR AFFILIATES BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES RELATING TO THE INFORMATION.

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation.

Contents

Appendix C

Appendix D

EnterpriseOne Order Promising 8.12.1 Preface

This preface discusses:

- Related documentation.
- Typographical Conventions and Visual Cues.

Note. This Implementation Guide documents only page elements that require additional explanation. If a page element is not documented with the process or task in which it is used, then it either requires no additional explanation or is documented with the common elements for the section, or chapter.

The *EnterpriseOne Order Promising 8.12.1Implementation Guide* provides you with information about how to implement and use your *EnterpriseOne Order Promising 8.12.1* system. However, additional essential information describing deployment and supplemental third party software options resides in the *EnterpriseOne Supply Chain Planning Hardware and Software Requirements Guide.* You should be familiar with the contents of this guide.

Documentation Updates and Downloading Documentation

This section discusses how to:

- Obtain documentation updates.
- Download documentation.

Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on Oracle's online support portal, Oracle MetaLink 3. Through the Documentation section of Oracle's online support portal, Oracle MetaLink 3, you can download files to add to your Implementation Guides Library. You'll find a variety of useful and timely materials, including updates to the full line of JD Edwards EnterpriseOne documentation that is delivered on your implementation guides CD-ROM.

Important! Before you upgrade, you must check Oracle's online support portal, Oracle MetaLink 3, for updates to the upgrade instructions. Oracle continually posts updates as the upgrade process is refined.

See Also

Oracle's online support portal, Oracle MetaLink 3, https://metalink3.oracle.com/od/faces/index.jspx

Downloading Documentation

In addition to the complete line of documentation that is delivered on your implementation guide CD-ROM, Oracle makes JD Edwards EnterpriseOne documentation available to you via Oracle's website. You can download PDF versions of JD Edwards EnterpriseOne documentation online via the Oracle Technology Network. Oracle makes these PDF files available online for each major release shortly after the software is shipped.

See Oracle Technology Network, http://www.oracle.com/technology/documentation/psftent.html.

Additional Resources

Additional resources are located on Oracle's online support portal, Oracle MetaLink 3: https://metalink3.oracle.com/od/faces/index.jspx

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.

Typographical Conventions

The following table contains the typographical conventions that are used in PeopleBooks:

Typographical Convention or Visual Cue	***Description***
Italics	Indicates field values, emphasis, and EnterpriseOne or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the number 0, not the letter O.
" " (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (|).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.

Typographical Convention or Visual Cue	*Description*
Cross-references	PeopleBooks provide cross-references either below the heading "See Also" or on a separate line preceded by the word See. Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Visual Cues

PeopleBooks contain the following visual cues.

Notes

Notes indicate information that you should pay particular attention to as you work with the EnterpriseOne system.

Note. Example of a note.

A note that is preceded by Important! is crucial and includes information that concerns what you must do for the system to function properly.

Note. Example of an important note.

Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

Note. Example of a warning.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other EnterpriseOne reference and training materials. Please send your suggestions to:

EnterpriseOne Product Documentation Manager EnterpriseOne, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@EnterpriseOne.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

About This Documentation

A companion Guide called *About This Documentation* contains general information, including:

- Related documentation, common page elements, and typographical conventions for guides.
- Information about using guides and managing the documentation library.
- Information on the International Organization for Standardization (ISO) country and currency codes used within documentation.
- A glossary of useful JD Edwards EnterpriseOne terms that are used in documentation.

You can find this companion PeopleBook in your PeopleSoft online library.

Chapter 1

Getting Started with Order Promising 8.12.1

This chapter provides an overview of Order Promising and discusses:

- Order Promising business process.
- Batch and real-time integration architecture
- Order Promising implementation

Order Promising Overview

Customers often need to receive immediate feedback about their orders. Using EnterpriseOne and Order Promising real-time integration, customers can be promised a specific delivery date at the time that their order is entered into the system. The Order Promising Server reviews the inventory, outstanding work orders, manufacturing routings, location of distribution centers, calendars, and company order-promising preferences to determine the feasibility of meeting a customer request date. Order Promising returns the order fulfillment results to EnterpriseOne for acceptance. The results also include the calculated costs associated with the available-to-promise (ATP), capable-to-promise (CTP), and profitable-to-promise (PTP) delivery of items. Configured items are supported for Capable-to-Promise (CTP). Available-to-promise (ATP) for configured items is only available through JD Edwards EnterpriseOne Availability Checking.

From EnterpriseOne Sales Order Entry, a customer service representative can automatically determine whether an order can be fulfilled for the order based on a selected service objective. If the initial fulfillment option is not acceptable to the customer, the customer service representative can select alternate service objectives until an acceptable result is found. Through Order Promising, you can optimize the internal costs of supplying your customers using available inventory, unused production capacity, or a combination of both. You can also make your inventory more flexible by reducing the need to consume expensive warehouse space with finished or semi-finished goods.

The Order Promising web application is also available to provide information about the simulated sales orders used for testing, available inventory, service objective definitions, allocated resources and administrative details.

Order Promising Business Process

The following process flow illustrates the Order Promising business processes:

1. The customer service representative enters order information in the EnterpriseOne Sales Order Entry program and submits the sales order inquiry for order promising using the Auto Promise option from the Form menu.

   Each customer is assigned a service objective in EnterpriseOne that corresponds to the service objectives configured in Order Promising. These service objectives instruct the Order Promising Server about how to fulfill the order. The customer service representative can override the service objective set for the customer, and choose a different service objective when entering the sales order. Any sales orders not associated with a service objective are promised using the Standard service objective.

2. EnterpriseOne sends the message containing the sales query information to the Order Promising Server.

3. The Order Promising Server receives a query message and attempts to fulfill the order based on its service objective. If the Order Promising server determines that the order can be manufactured based on the associated service objective, manufacturing algorithms are applied.

   If proximity searching has been selected for the customer, the Order Promising server converts the customer's location into geographic coordinates and then uses the Proximity Search algorithm to determine the locations that can most effectively fulfill this order. If preferred sourcing has been selected for this customer, Order Promising attempts to fulfill the order from the chosen locations.

4. The Order Promising Server returns the results of the query to EnterpriseOne Sales Order Entry including details about how Order Promising has fulfilled the order.

5. The customer service representative can commit the order if the customer is agreeable to the proposed fulfillment option. Alternatively, they can resubmit the customer order using a different service objective or change the customer request date or quantity.

6. EnterpriseOne sends the committed order in real time to Order Promising. In addition, changes to sales orders, purchase orders, transfer orders, and manual inventory adjustments are communicated to Order Promising in real-time. For non-configured items, Order Promising first consumes available inventory (ATP), and then manufacturing capacity or resources (CTP) if applicable.

Order Promising Batch and Real-time Integration Architecture

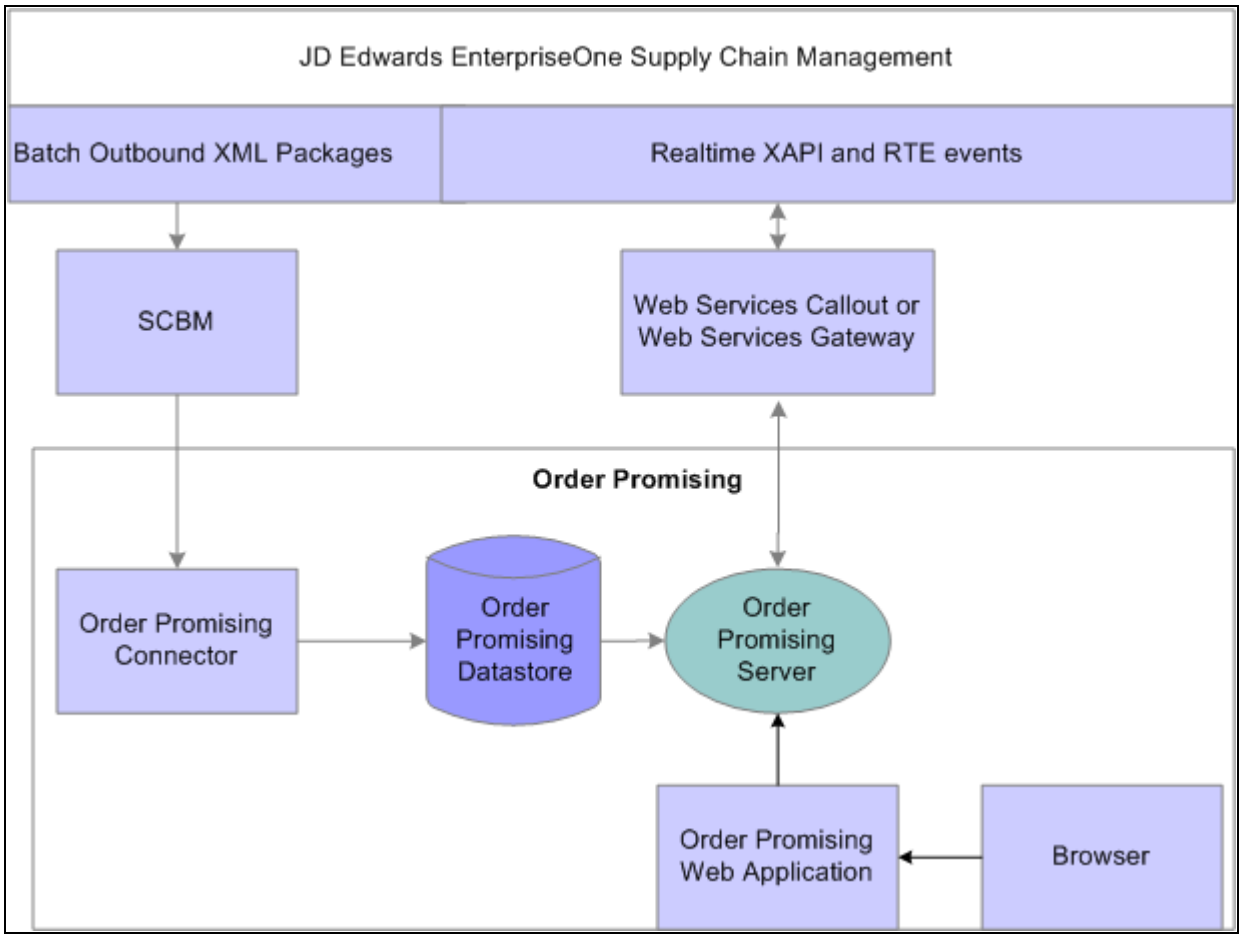
Order Promising supports two forms of integration with EnterpriseOne:

- Batch
- Real-time

There are two realtime integration approaches available:

- Web Services Gateway: uses WebMethods to transmit realtime (RTE) and XAPI events.
- Web Services Callout: uses business services to consume the Order Promising web service. This option is only available with EnterpriseOne Tools 8.97.

The following process flow illustrates how batch and real-time data moves between EnterpriseOne and Order Promising:



Flow of batch and real-time data between EnterpriseOne and Order Promising

EnterpriseOne outputs batch extracts that contain key Supply Chain information in XML format. Batch integration is fundamental and required to load the Order Promising model. The Supply Chain Business Modeler transforms the EnterpriseOne batch extracts, which the Order Promising Connector then uses to directly update the Order Promising datastore. If you are going to promise orders in real time, it is recommended that batch routines also be run on a regular basis to synchronize the EnterpriseOne and the Supply Chain Planning Order Promising models. In particular, there are some events not published during real time such as manufacturing information, transportation, and customer information.

Real-time notification messages allow JD Edwards EnterpriseOne sales orders containing either standard or configured items to be promised by the Order Promising Server for a specific date. Any changes in sales orders, purchase orders, transfer orders, work orders, and manual inventory adjustments are communicated to the Order Promising in-memory model as they occur. This communication ensures that the information that Order Promising uses to fulfill orders is up-to-date.

Note. Real-time notification messages update the in-memory model. Details within the Order Promising datastore are updated from the RequestJournal file when the Order Promising server is restarted.

Batch Integration Process

The batch extract processor can be run either manually, from the Supply Chain Planning command line, or by the EnterpriseOne Scheduler, which automates the scheduling. EnterpriseOne creates these XML packages that can be exported into the Supply Chain Business Modeler:

- Base
- Beginning Inventory
- Customer
- Distribution
- Manufacturing
- Purchase Orders
- Sales Orders
- Supplier
- Transfer Orders
- Work Orders

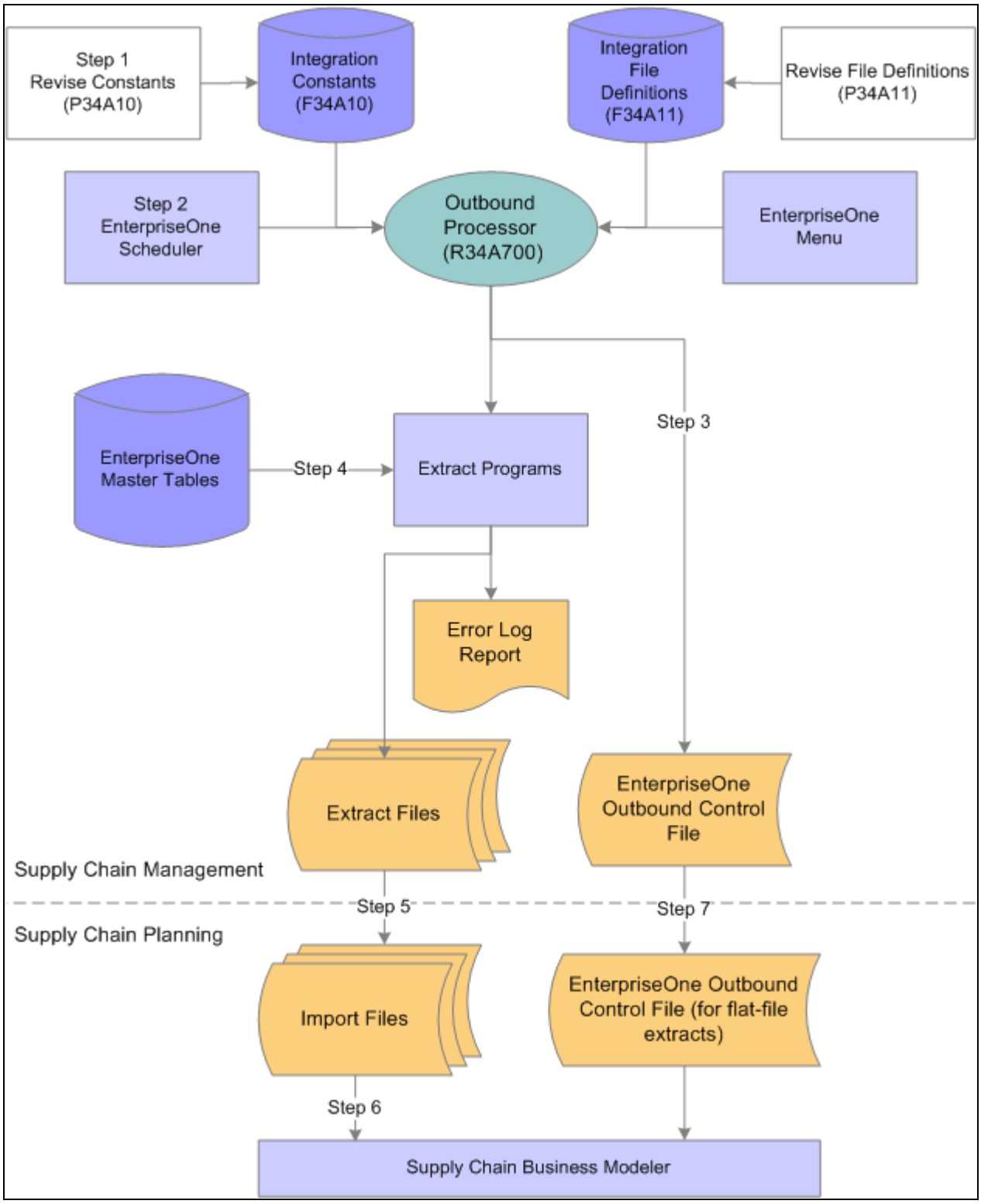
The SCBM Outbound Processor (R34A700) is used to export XML packages.

The system flow for outbound integration from Supply Chain Management to Supply Chain Planning is:

1. Set up integration constants and file definitions using an interactive application.
2. Launch the outbound processor either through the EnterpriseOne Scheduler, from a menu option, or through the RunUBE command from the Supply Chain Planning command line.
3. If batch processing is enabled, the system verifies that the previous batch job has processed.
4. The system calls the extract programs that you specified in the processing options.
5. The extract programs create the XML files that you requested and transfers them directly to the Supply Chain Business Modeler extract directory.
6. The Supply Chain Business Modeler imports the EnterpriseOne XML extract files from the extract directory and transforms the data into the level of detail required for a tactical planning application like Order Promising.

   Finally, the Supply Chain Business Modeler exports the data from the Tactical Model.
7. The Order Promising Connector imports the data from the SCBM Tactical Model and transforms the data.
8. The Order Promising Connector updates the Order Promising datastore.

This flowchart illustrates the preceding steps:



Order Promising Batch Integration with EnterpriseOne Supply Chain Management

Real-time Integration Process using Web Services Callout

This section provides an overview of the real-time sales order query and model updating processes using Web Services Callout (WSC). The Web Services Callout realtime integration approach uses business services to consume the Order Promising web service. When the system transmits sales order, purchase order, manual inventory, work order, and work order parts list and routing events from EnterpriseOne to Order Promising, the Order Promising business service converts the events into the xml format required by the Order Promising Server.

Sales Order Query Process

These actions are associated with the processing of the EnterpriseOne events involved in sales order queries:

1. The customer service representative initiates order promising from the form exit using Auto Promise. When the customer service representative activates the order promising option, the sales order query is committed to RTE memory, and the Order Promising business service commences. A "processing" message is displayed.

2. The Order Promising business function retrieves the sales order query from RTE memory, builds an xml message, then calls the business service that transfers the message to the Order Promising web service.

3. The Order Promising web service transfers the sales query to the Order Promising server.

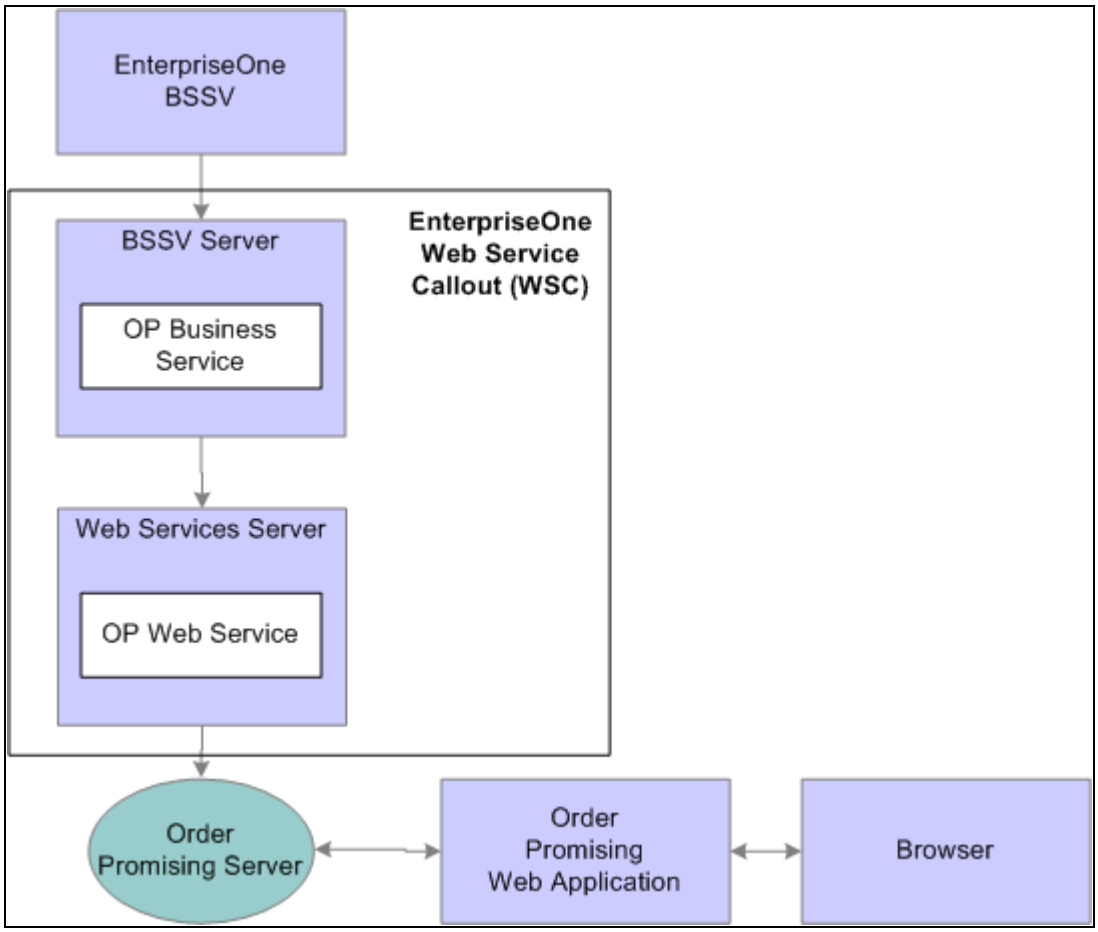
4. Using the service objectives and other promising configurations, the Order Promising server determines whether the order can be fulfilled by the customer's requested date and returns order fulfillment details.

5. The Order Promising server sends the response to the Order Promising web service.

6. The Order Promising web service transfers the incoming the response message to the Order Promising business service where it is formatted for use by EnterpriseOne.

7. The EnterpriseOne Sales Order Entry program displays the results or errors. The representative can either commit the result or restart the order in a prepromised state. If the customer service representative commits the order, the order details are transferred to the Order Promising Server, which updates the Order Promising in-memory model with the commitment, and allocates the necessary inventory and resources. If the customer is not satisfied with the order fulfillment details returned by Order Promising, the customer service representative can choose a different service objective, and rerun the query.

The following flow chart illustrates how sales order query events are processed between EnterpriseOne and Supply Chain Planning using the Web Services Callout approach:



Sales Order Query Process

Model Update Process

The following steps are associated with the processing of EnterpriseOne events to update the Order Promising in-memory model:

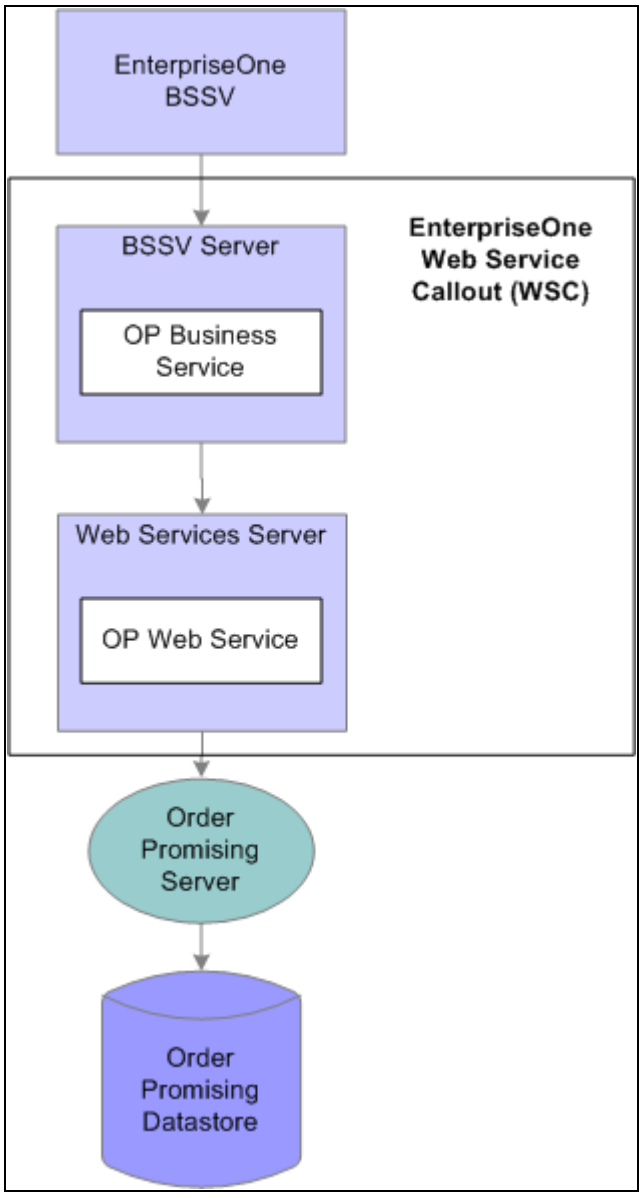
1. When a change to a sales order, work order, parts list or routing, inventory, lot status, purchase order or transfer order occurs, the application calls a business function which saves the details in the RTE memory, then calls another business function that builds an xml message from the event data and passes that data to the OrderPromising Processor (J34A0010) business service.

   The following business services may be called depending on the type of change being transmitted:

Type of Event	*Call Name*	*Business Function*
Inventory Adjustment	callOPAdjustInventoryBSSV	B34A1310
Purchase or Transfer Order	callOPProcurementBSSV	B34A1300
Sales Order	callOPSalesBusinessService	B34A1370
Work Order	callOPWorkOrderBusinessService	B34A1320

2. The Order Promising business service retrieves the location where the Order Promising web service is deployed, maps the data it has just received from the business function and calls the web service.
3. The Order Promising web service transfers the model update to the Order Promising server.
4. The Order Promising server updates the in-memory promising model.
5. The Order Promising model datastore is updated upon completion of the Order Promising server session.
6. The Order Promising server returns a success message or an error code to the business service. The business service reads the message and either returns a success or cross referenced error code, which is returned back to the error recovery application.

The following flow chart illustrates how the model update events are processed between EnterpriseOne and Supply Chain Planning using the Web Services Callout approach:



Model Update Process

Real-time Integration Process using Web Services Gateway

This section provides overviews of the real-time sales order query and model updating processes using the Web Services Gateway, an approach that uses webMethods and integration points to transmit, format and map messages between EnterpriseOne and Order Promising.

Sales Order Queries

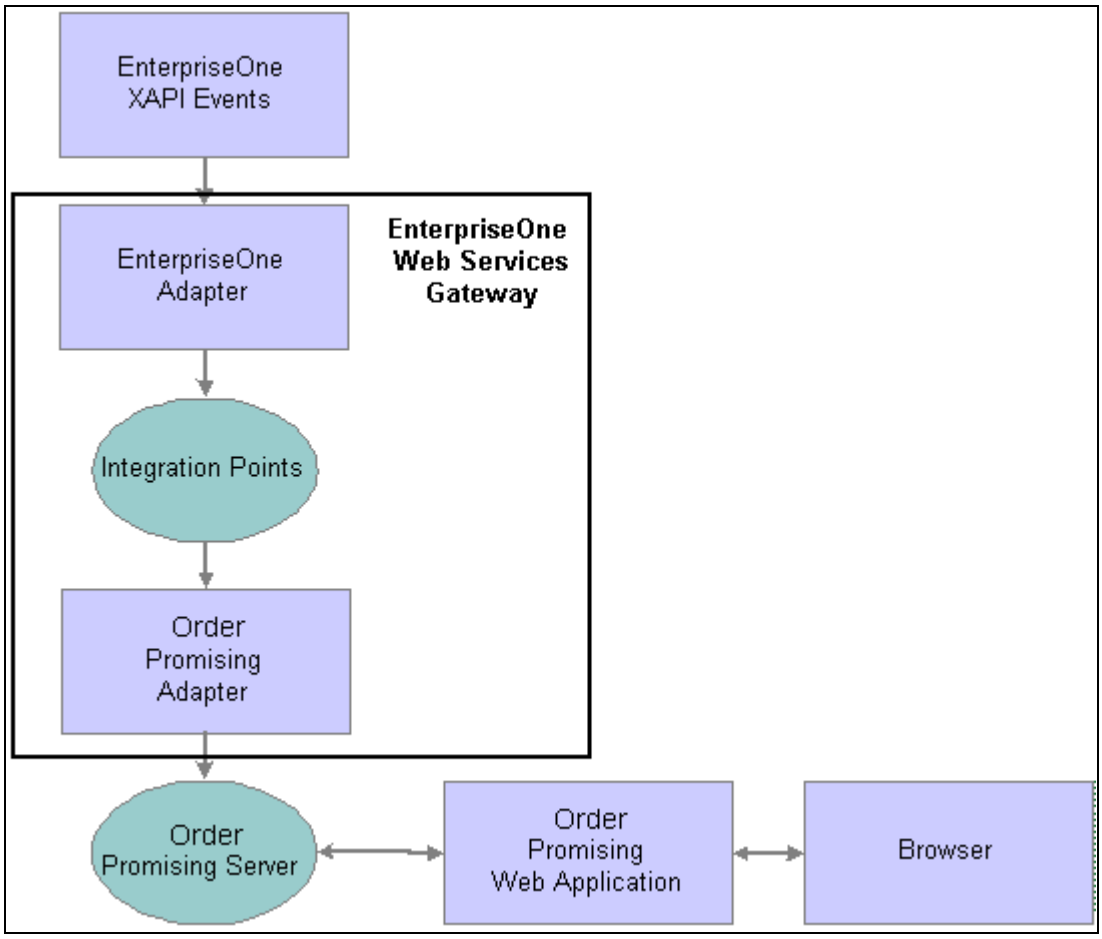
In real-time integration, when new sales orders are promised, EnterpriseOne immediately sends a query to Order Promising. Order Promising can promise both standard and configured sales orders. Configured sales orders are created when a customer service representative specifies the features and options for each item by using the Configurator program, which is started from within the Sales Order Entry program. For each configured item, the Configurator program generates a unique manufacturing routing and parts list, which needs to be communicated to Order Promising. These real-time events support order promising for both standard and configured items:

- notifySalesOrderPromise
- notifySalesOrderResponse

Note. In addition, the notifyWorkOrdersand notifyWorkOrderBOMR are sent in real time for any configured sales orders.

The real-time query does not consume any inventory, capacity or resources until the order is committed. After the order is committed, the Order Promising in-memory model is updated with the sales order information.

This process flow diagram shows how the EnterpriseOne events are processed between EnterpriseOne and Order Promising with the Order Promising web application monitoring the Order Promising Server:



Flow of real-time order promising messages between EnterpriseOne and Order Promising

These actions are associated with the processing of the EnterpriseOne sales order queries:

1. The customer service representative creates a sales order and selects a service objective for fulfilling the order. If no service objective is selected for the sales order, the service objective associated with the customer is used. When a service objective is not associated with the customer, the Standard service objective is assigned.
2. The customer service representative initiates order promising from the form exit using Auto Promise, sending the sales order query from EnterpriseOne to the EnterpriseOne adapter.
3. Integration points convert the data from the EnterpriseOne adapter to the format supported by the Order Promising Server, and transfer the sales query message to the Order Promising adapter.
4. The Order Promising adapter transfers the sales query message to the Order Promising Server. Based on the service option assigned to the customer, the best fulfillment option is computed.
5. The Order Promising Server sends the response to the Order Promising adapter.
6. Integration points convert the response message to the format used by EnterpriseOne, and send the data to the EnterpriseOne adapter.

7. The EnterpriseOne adapter sends the data to the EnterpriseOne Sales Order Entry program.

   The customer service representative uses the form exit to return to EnterpriseOne. From there, the representative can either commit the result or return the order to its prepromised state and rerun the query. If the customer service representative commits the order, the order details are transferred to the Order Promising Server, which updates the Order Promising in-memory model with the commitment. The Order Promising Server then allocates the necessary inventory and resources.

See "Appendix A: Understanding the Mappings"

Model Updating

The Order Promising in-memory model is updated by EnterpriseOne in real time when:

- Customer service representatives commit the fulfillment option in the EnterpriseOne Sales Order Entry program, thereby transferring order details.
- Customer service representatives update a committed sales order.
- Service personnel on the shop floor modify a purchase order, transfer order, or manually adjust the inventory.

Note. For both production and maintenance work orders, these fields must be complete to successfully publish an order to Order Promising: Request Date, Order Status, Order Type, and Branch Plant.

These messages transfer information from EnterpriseOne to Order Promising:

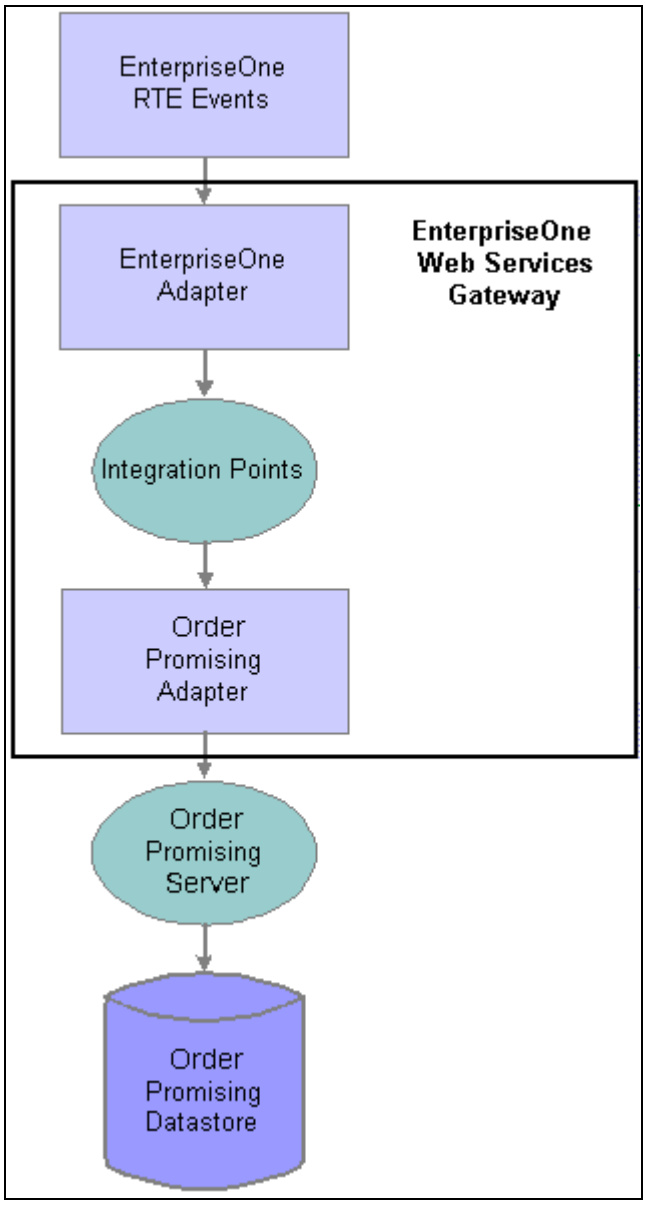
- notifyItemBalance
- notifyPurchaseTransferOrder
- notifySalesOrder
- notifyWorkOrder
- notifyWorkOrderBOMR

Note. The notifyPurchaseTransferOrder contains both purchase order and transfer order information.

Note. When sales orders are committed, the notifySalesOrder message is sent from EnterpriseOne to Order Promising in real time. In addition, non-standard sales orders containing configured items send the notifyWorkOrder and notifyWorkOrder BOMR messages to Order Promising in real time.

Other changes to work orders, parts lists and routings are reflected in the Order Promising model after the Order Promising server is restarted.

This flowchart shows how the EnterpriseOne messages are processed between EnterpriseOne and Order Promising:



Flow of real-time model updating messages between EnterpriseOne and Order Promising

The following steps are associated with the processing of EnterpriseOne messages to update the Order Promising in-memory model:

1. When a change to a sales order, manual inventory adjustment, purchase order or transfer order occurs, EnterpriseOne sends the details to the EnterpriseOne adapter.
2. Integration points convert the message from the EnterpriseOne adapter to the SCP format supported by the Order Promising Server, and transfer the message to the Order Promising adapter.
3. The Order Promising adapter transfers the message to the Order Promising Server.
4. The Order Promising Server updates the Order Promising in-memory model.
5. The Order Promising datastore is updated when the Order Promising server session is ended.

See "Appendix A: Understanding the Mappings"

Order Promising Implementation

The Order Promising implementation can be divided into these phases:

- Setting up a data model.
- Configuring the Order Promising system.
- Setting up real-time integration between EnterpriseOne and Order Promising.

In the planning phase of your implementation, take advantage of all EnterpriseOne sources of information, including the installation guides and user guides. A complete list of these resources appears in the preface, with information about where to find the most current version of each.

Setting Up the Data Model

The steps discussed in this section define information in your Order Promising data model. The information that you define in the datastore is used as the foundation for all order promises.

Step	***Reference***
Set EnterpriseOne processing options to extract batch files.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Defining General Integration Settings"* *Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Setting Up the SCBM Outbound Processor (R34A700)"*
Extract the EnterpriseOne batch extracts to the Supply Chain Business Modeler extract directory.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Transferring Data Between EnterpriseOne and Supply Chain Planning"*
Set the Supply Chain Business Modeler to import the EnterpriseOne extracts.	*JD Edwards EnterpriseOne Supply Chain Business Modeler 8.12.1 User Guide, "Understanding Data for Importing Into and Exporting From Supply Chain Business Modeler"*
Transform the EnterpriseOne data into the level of detail required for a tactical planning application.	*JD Edwards EnterpriseOne Supply Chain Business Modeler 8.12.1 User Guide, "Setting Up Models"*
Export the data in the Tactical Model from the Supply Chain Business Modeler.	*JD Edwards EnterpriseOne Supply Chain Business Modeler 8.12.1 User Guide, "Exporting Data from Supply Chain Business Modeler"*
Run the Order Promising Connector to update the Order Promising datastore with the data from the SCBM Tactical Model.	*"Using the Order Promising Connector"*

Configuring the Order Promising System

The steps discussed in this section enable you to configure the default Order Promising parameters, geographic aliases, and service objectives that govern the functioning of the Order Promising program.

Step	Reference
Configure the Order Promising server variables.	*"Configuring the Promising Server"*
Configure the default Order Promising datastore variables.	*"Setting Datastore Configuration Variables"*
Add any additional geographic aliases.	*"Defining Geographic Aliases"*
Create service objectives.	*"Defining Service Objectives"*
Create any resource allocations.	*"Allocating Resources"*
Test the functioning of the service objectives with a sample sales order to verify the effectiveness of your service objectives.	*"Simulating Sales Order Processing"*

Setting Up Real-time Integration between EnterpriseOne and Order Promising using Web Services Callout

The steps discussed in this section provide the information necessary for configuring a real-time integration with EnterpriseOne Sales Order Management using Web Services Callout (business services that consume the Order Promising web service).

Step	Reference
Install and configure real-time EnterpriseOne integration components.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Setting Up JD Edwards EnterpriseOne to Integrate with Supply Chain Planning using the Web Services Callout Approach"* *JD Edwards EnterpriseOne Order Promising 8.12.1 Installation Guide*
Configure EnterpriseOne UDC tables and processing options for real-time integration.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Defining General Integration Settings"* *Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Setting Up Realtime Order Promising"*
Install and start the Order Promising Server.	*JD Edwards EnterpriseOne Order Promising 8.12.1 Installation Guide* contains information about how to install and configure Order Promising.
Create and send a sales order to Order Promising.	*JD Edwards EnterpriseOne EnterpriseOne 8.12 Sales Order Management User Guide, "Generating Delivery Proposals with Order Promising"*

Setting Up Real-time Integration between EnterpriseOne and Order Promising using the Web Services Gateway

The steps discussed in this section provide the information necessary for configuring a real-time integration with EnterpriseOne Sales Order Management using the Web Services Gateway (webMethods and integration points).

Step	***Reference***
Install and configure real-time integration components.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Setting Up JD Edwards EnterpriseOne to Integrate with Supply Chain Planning using the Web Services Gateway Approach"*
Configure EnterpriseOne UDC tables and processing options for real-time integration.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Defining General Integration Settings"* *Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Setting Up Realtime Order Promising"*
Start and configure the EnterpriseOne adapter.	*JD Edwards EnterpriseOne 8.96 Web Services Gateway Installation and Setup Guide (Microsoft Windows and UNIX Systems), "Configuring the JD Edwards EnterpriseOne Adapter"*
Start and configure the Order Promising adapter.	*JD Edwards EnterpriseOne 8.96 Web Services Gateway Installation and Setup Guide (Microsoft Windows and UNIX Systems), "Configuring the JD Edwards EnterpriseOne Order Promising Adapter"*
Install the Order Promising integration points.	*Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Reviewing the Installation and Configuration Steps"* *JD Edwards EnterpriseOne 8.12 SP1 Integration Points Installation (Microsoft Windows and UNIX Systems)*
Install and start the Order Promising Server.	*JD Edwards EnterpriseOne Order Promising 8.12.1 Installation Guide for Windows/UNIX* contains information about how to install and configure Order Promising.
Create and send a sales order to Order Promising.	*JD Edwards EnterpriseOne 8.12 SP1 Sales Order Management User Guide, "Generating Delivery Proposals with Order Promising"*

Chapter 2

Understanding Order Promising Components

This chapter discusses Order Promising's major components. They include:

- Order Promising Server
- Order Promising data
- Order Promising Web application
- Order Promising Data Connector
- EnterpriseOne integration components

Order Promising Server

The Order Promising Server receives EnterpriseOne real-time messages from either the EnterpriseOne Web Services Gateway or the web server depending on webMethods or business services are being used to transfer messages. The Order Promising Server compares the real-time sales order requests against the Order Promising model of your supply chain that resides in memory. Throughout the order promising cycle, this representation is updated to reflect the changing demands and constraints that affect your supply chain. When the Order Promising server performs queries, it tries to allocate the available inventory and capacity to a sales order inquiry. Based on the configuration and service objectives selected by your organization, and the requested delivery date, the Order Promising server returns the best fulfillment option to the EnterpriseOne Sales Order Entry program.

Order Promising provides three sophisticated functions that can be used with your external systems to determine the most effective delivery date for sales orders based on the service objectives set by your organization. They are:

- Available to Promise
- Capable to Promise
- Profitable to Promise

Available to Promise

For non-configured items, Available to Promise (ATP) checks projected inventory levels across the supply chain for a defined planning horizon. Order Promising uses ATP functionality to return the best fulfillment solution for your customers using available inventory, while respecting business constraints and minimizing cost.

Capable to Promise

Capable to Promise (CTP) extends ATP by checking manufacturing constraints and costs. If inventory is not available, CTP determines when the inventory can be produced. When determining the fulfillment solution for your customers, CTP considers:

- Production capacities
- Material availability (or material manufacturing time)
- Subassemblies (material availability, manufacturing time, costs)

For non-configured items, an ATP timefence can be assigned on an item-by-item basis. Orders for an item with an ATP timefence requested after the set date will be produced by CTP, and not exhaust the available inventory. For more information about setting the ATP timefence, see *Defining an ATP Timefence.*

Profitable to Promise

Profitable to Promise (PTP) allows you to maximize the profitability of fulfilling an order. In Order Promising, costs are determined by the unit cost, the distance and cost involved in shipping the unit, and the cost of manufacturing the item if required. Service objectives can be configured to instruct the server to search for the most profitable solutions. You can also configure the fulfillment rules associated with each service objective to emphasize less expensive approaches to shipping, manufacturing, and sourcing. The cost and profit details are displayed with the fulfillment solution.

Order Promising Data

The Order Promising data component includes:

- Order Promising model and configuration datastore
- Geographic database

These two forms of data interact to provide the information Order Promising requires to fulfill orders.

Understanding the Order Promising Model and Configuration Datastore

The Order Promising datastore is the repository for the Order Promising model, configuration, and service objective information. It is loaded into memory when the Order Promising server is started to facilitate rapid searching and quick response to queries. To maximize promising performance, the data representation on disk is not updated with the real-time messages until the Order Promising server is properly stopped. To protect against any loss of data, every message from EnterpriseOne is also recorded in the .requestJournal file. In the case of system failure, the .requestJournal file updates the datastore on disk when the Order Promising server is restarted.

The Order Promising datastore, formatted in XML, is updated in four ways:

- EnterpriseOne batch extracts that supply SCBM, and then Order Promising
- Real-time changes to the EnterpriseOne Supply Chain Management model

- Order Promising web application changes
- Manual XML changes

Note. File locking on the Order Promising datastore restricts access at any given time by either one session of the Order Promising server or the Order Promising connector.

EnterpriseOne Batch Extracts

The Order Promising datastore model contains supply chain management information from EnterpriseOne including:

- Beginning Inventory
- Calendar
- Customer
- Manufacturing
- Purchase Order
- Supplier
- Transfer Order
- Work Order

The EnterpriseOne supply chain management information is initially sent to the Supply Chain Business Modeler, which augments and reformats the data, finally exporting the data in the form of the Tactical Model. The Tactical Model is loaded into the Order Promising datastore after being transformed by the Order Promising Connector.

Real-time Changes to the EnterpriseOne Supply Chain Management Model

Changes in EnterpriseOne sales orders, purchase orders, transfer orders, manual inventory, lot status and work orders and their parts list and routings are communicated to Order Promising in real time. These events are sent directly to the Order Promising server by EnterpriseOne Business Services which updates the in-memory promising model. These events are recorded in the requestJournal, a file that logs all the real-time messages received from EnterpriseOne. This file updates the datastore with all the real-time EnterpriseOne messages received during the current session when the Order Promising server is stopped. When the Order Promising server is restarted, the contents of the datastore are loaded into the in-memory promising model and used to promise subsequent real-time orders.

Standard work orders, work order parts list and routing events are also sent to the Order Promising server in real time but unlike the other real-time events, they do not update the in-memory promising model. They are recorded in the .requestJournal, a file that logs all the real-time messages received from EnterpriseOne. This file updates the datastore with all the real-time EnterpriseOne messages received during the current session when the Order Promising server is stopped. When the Order Promising server is restarted, the contents of the datastore are loaded into the in-memory promising model and used to promise subsequent real-time orders.

Order Promising Web Application Changes

The Order Promising web application allows you to create, modify and delete service objectives used by Order Promising to determine how orders are fulfilled. In addition, you can simulate a sales order and promise it in Order Promising. All changes made to service objectives and simulated sales orders in the Order Promising web application are stored in the datastore at the end of the current session.

Manual XML Changes

Some of the datastore objects can only be updated manually in XML. They include:

- Order Promising server variables

  These variables configure the Order Promising server by specifying the port, datastore directory, schema directory, log file, geographic data file, reports, and troubleshooting options.

- Datastore variables

  These variables determine the functioning of the promising function, such as the horizon start time and length, item rounding, material procurement, and other integration information.

- Resource allocation data

  The information required to allocate resources in the future for the production of a specific item for a customer.

- Geographic aliases

  The different sales order city, state, and country spelling combinations that are acceptable to the geographic database, ensuring the proper sourcing of items and materials, and the determination of order costs and profitability.

Note. Although Order Promising allows you to access the XML documents when the Order Promising server is running, you will not be able to save the changes until the Order Promising server has been stopped. This is because the XML documents are overwritten by the EnterpriseOne real-time messages when the Order Promising server is shutdown, therefore any changes made during the session will become obsolete.

See Also

"Appendix B: Understanding the Supply Chain Planning XML Format"

"Creating Allocation Contracts"

"Restoring the Real-time EnterpriseOne Messages to the Datastore"

"Appendix C: Understanding the Order Promising XML Format"

"Defining Service Objectives"

"Simulating Sales Order Promising"

"Configuring Server Variables"

"Setting Datastore Configuration Variables"

"Defining Geographic Aliases"

JD Edwards EnterpriseOne Supply Chain Business Modeler 8.12.1 User Guide, "Exporting Data from Supply Chain Business Modeler"

Integrating JD Edwards EnterpriseOne 8.12 with Supply Chain Planning, "Setting Up the SCBM Outbound Processor (R34A700)"

Geographic Database

The geographic database contains information about approximately 2.85 million cities, towns, and villages, along with their geographic coordinates. When an order is entered for a specific customer, Order Promising initiates a proximity search using the city specified in the sales order. During the proximity search, the Order Promising server converts addresses to longitude and latitude to calculate distance, delivery cost per unit ordered, and lead times for shipping and arrival dates. The geographic database also enables Order Promising to determine the best plant or distribution center to fulfill an order.

Order Promising Web Application

The Order Promising Web Application provides customers with five tabs that can be used to configure, test and review order promising settings:

- Simulated Sales Orders
- Service Objectives
- Allocation Manager
- Available Inventory
- Administration

The Order Promising application is deployed on a web application server and accessed through a browser.

Simulated Sales Orders Tab

The Sales Orders tab allows you to simulate and verify the promising of orders by the Order Promising Server once the service objectives and fulfillment rules have been configured. From this tab, you can create, edit, duplicate and delete sales orders and their line items. Each sales order can be linked with a specific service objective, and can be configured to either allow or disallow partial order shipments, backorders, partial line item shipments, multisourcing, or product substitutions. After reviewing the promised results, you can return to the sales order, make changes, and then repromise until you achieve the results you want.

Service Objectives Tab

A large global enterprise can have a vast supply chain. EnterpriseOne Order Promising considers this situation and can determine thousands of possibilities to fulfill orders. The Order Promising server tailors the possibilities using service objectives. These service objectives help EnterpriseOne Order Promising to provide answers that reflect the methods in which you fulfill sales orders.

You might, for example, have inventory at different distribution centers around the world, which would normally never be promised to a particular order in North America. You might also have premium manufacturing capacity in your supply chain that could be used when economies of scale dictate. However, you would never promise orders based on scheduling overtime, when the order might be available from regular production capacity or from the available inventory supply on the next day.

The Service Objectives tab provides you with the capability to configure and compare service objectives. Multiple service objectives can be created to differentiate the level of service offered to your customers when the order is being promised, thereby affecting how the order is fulfilled. You can configure your service objectives to offer better service to your most important customers or maximize profitability. You can specify both the level of service that a customer receives and the preferred sources of supply.

Although each customer is assigned a default service objective, this default can be overridden when the order is promised by the customer service representative.

Service objectives are made up of one or more fulfillment rules that the Order Promising Server uses to determine how to fulfill the order. Fulfillment rules can be created in these categories:

- Logistics
- Manufacturing
- Delivery
- Product Substitution
- Sourcing

Logistics Rules

Logistics data helps you manage the internal distribution processes of your enterprise according to the applicable service objective used when promising an order. You can open and close item or item group-specific shipping lanes and transport modes with effective dates or a horizon timefence. You can assign rules to specific customers or groups of customers, which allows you to condense the full distribution topology.

Manufacturing Rules

You can leverage the power of EnterpriseOne Order Promising Capable to Promise (CTP) functionality by allowing or disallowing manufacturing processes at specific locations or groups of locations. You can allow or disallow manufacturing processes at specific locations for a specific customer, a customer group, an item, or an item group. You can make routings unavailable, or allow or disallow premium manufacturing capacity. You can specify a non-preferred manufacturing routing or non-optimal processes for promising. If your enterprise has constrained capacities, or if specific processes are unavailable due to maintenance, you can use this routing or process as a backup.

Delivery Rules

Delivery rules can determine the final availability date and delivery cost to the customer. Within a specified lane, you can specify a fixed lead time, or a lead time that varies based on the distance the item is being shipped. You can also specify the shipping cost by weight.

Product Substitution Rules

EnterpriseOne Order Promising allows you to specify the item to substitute when the customer's primary choice is unavailable. You can define substitution ratios and costs per unit of substitution. Product substitutions can be performed for individual items and across entire groups of items.

Sourcing Rules

Creating sourcing rules allows you to influence the sourcing method for specific customers or customer groups for an item or item group. EnterpriseOne Order Promising allows you to source an item by its proximity to the customer or by preferred sourcing.

When you enable proximity searching, EnterpriseOne Order Promising can automatically determine the closest locations to the shipping address on the sales order by using a global database of 2.85 million cities, towns, and villages along with their geographic coordinates. Up to five locations can be found and displayed sorted by delivery cost, distance, or lead time. When an order is received for a customer using proximity sourcing, Order Promising converts the source and destination into geographic coordinates to calculate the shipment distance, delivery cost per unit ordered, and lead time.

Alternatively, Order Promising can query up to four preferred sources when promising orders for specific customers or items.

Allocation Manager Tab

The Allocation Manager tab allows you to view resource allocations associated with specific items. The resource allocations are administered through the use of customer allocation contracts which specify the amount of resources or items to be reserved for the customer, time period, and location. The Allocation Manager tab also provides details of the customer contracts such as the start and end dates and details about when the reservation expires.

Available Inventory Tab

The Available Inventory tab allows you to check the quantity available of specific items throughout your organization, or at specific locations. Order Promising provides a snapshot of your inventory over a two week period of time, and provides a breakdown of the item at each location. This information can help you to understand inventory availability if your customer challenges the fulfillment results generated by Order Promising.

Administration Tab

The Administration tab provides tools that you can use to:

- Check the Order Promising server status for the current session. Details include the total number of promises, the average promise time, the number of protocol and promising errors, and the slowest promise time.
- View the horizon, logging, and system configuration settings.
- View the sales order queries in the queue, and how long they have been waiting.
- View information about the slowest promises during the current session including details about the sales order number, the customer, the number of line items, the customer service representative, and the amount of time it took to promise the order.

Order Promising Data Connector

Using a connector that is provided with EnterpriseOne Order Promising, you can transfer enterprise data from EnterpriseOne Supply Chain Business Modeler to EnterpriseOne Order Promising. EnterpriseOne Supply Chain Business Modeler is a configurable supply chain data warehouse that enables you to transfer enterprise data between EnterpriseOne supply chain management and supply chain planning systems.

After importing supply chain data into EnterpriseOne Supply Chain Business Modeler, you can export the data from the Tactical model for use in EnterpriseOne Order Promising. Using the data connector, you can convert the data into an import file and load the data into EnterpriseOne Order Promising.

See Also

"Using the Order Promising Connector"

EnterpriseOne Web Services Callout Integration Components

To integrate with the EnterpriseOne Sales Order Management system using business services, a number of integration components must be used to integrate with Order Promising. These components include:

- Order Promising Business Service

- Order Promising Web Service

Order Promising Business Service

The Order Promising business service is deployed on the Business Services Server. This business service retrieves the location of the OP web service and calls the web service passing data from EnterpriseOne. The response it gets from the web service is transformed and returned to the calling business function.

Order Promising Web Service

Web services enable software applications that are written in various programming languages and are running on various platforms to exchange data over computer networks.

The Order Promising web service is deployed on an application server. It takes the data sent by the business service and sends it directly to the Order Promising server.

EnterpriseOne Web Services Gateway Integration Components

To integrate with EnterpriseOne Sales Order Management system, a number of additional integration components must be used to integrate with Order Promising. These components are installed with the EnterpriseOne Web Services Gateway. They include:

- EnterpriseOne Integration Server
- EnterpriseOne Adapter
- Order Promising Adapter
- EnterpriseOne to Order Promising Integration Points

EnterpriseOne Integration Server

The EnterpriseOne Integration Server is the technology used to transmit information between EnterpriseOne and Order Promising. This technology provides a process for transmitting data with completely different formats from one environment to another.

When the system transmits sales order, purchase order, manual inventory, work order, and work order parts list and routing events from EnterpriseOne to Order Promising, EnterpriseOne Integration Server integration points convert the events into the format required by the Order Promising Server.

The EnterpriseOne Integration Server enables the exchange of data and logic by serving as an enterprise-wide integration backbone. Resources that you want to integrate connect to this integration backbone instead of directly to each other. The Integration Server performs the essential work of transporting information among resources, dispatching documents according to established business rules, and invoking processes on target systems. It also hosts integration logic, performs data transformation, and supports both synchronous (RPC and request/reply) and asynchronous (messaging) modes of interaction among resources.

For the real-time order promising function, real-time events are generated by EnterpriseOne, and then passed through the EnterpriseOne Adapter, the EnterpriseOne to Order Promising integration points, and finally, the Order Promising Adapter before reaching the Order Promising Server. The adapters and integration points are hosted or deployed to the EnterpriseOne Integration Server.

See Also

EnterpriseOne 8.95 Web Services Gateway Installation and Setup Guide (Microsoft Windows and UNIX Systems)

EnterpriseOne 8.95 Web Services Gateway Integration Developer User's PeopleBook

Order Promising Adapter

The Order Promising adapter transfers real-time messages between the integration points and the Order Promising Server. The Order Promising adapter must be configured to publish and listen for real-time messages from both the EnterpriseOne adapter and the Order Promising Server.

See Also

EnterpriseOne Tools 8.95 Web Services Gateway PeopleBook: Order Promising Adapter Programmers Guide

EnterpriseOne 8.95 Web Services Gateway Installation and Setup Guide (Microsoft Windows and UNIX Systems)

EnterpriseOne Adapter

This component transmits events from EnterpriseOne to the Order Promising adapter. In addition, it receives messages from Order Promising via the Order Promising adapter. This adapter works in combination with the EnterpriseOne Integration Server. The EnterpriseOne Adapter must be configured to publish and listen for messages from EnterpriseOne. It must also be configured to publish and listen for messages from the Order Promising adapter.

The EnterpriseOne adapter transfers real-time messages between the EnterpriseOne and the integration points. The EnterpriseOne adapter must be configured to publish and listen for real-time messages from both the EnterpriseOne and the Order Promising Adapter.

See Also

EnterpriseOne Tools 8.95 Web Services Gateway PeopleBook: EnterpriseOne Adapter Programmer's Guide

EnterpriseOne 8.95 Web Services Gateway Installation and Setup Guide (Microsoft Windows and UNIX Systems)

EnterpriseOne to Order Promising Integration Points

Integration points are used to convert data from the EnterpriseOne format to the SCP format required by Order Promising, and back to the EnterpriseOne format. In addition, the integration points filter the real-time messages being published by the EnterpriseOne adapter so that only those messages that are meant for planning purposes are sent through to the Order Promising Adapter, and subsequently the Order Promising Server.

See Also

EnterpriseOne 8.12.1 SP1 Integration Points Installation (Microsoft Windows and UNIX Systems)

Chapter 3

Working With EnterpriseOne Order Promising

This chapter discusses how to:

- Start the EnterpriseOne Order Promising server.
- Sign in to EnterpriseOne Order Promising.

Starting the Order Promising Server

This section discusses how to:

- Start the server in Windows
- Start the server in UNIX

Note. Before starting the Order Promising server, make sure that no other sessions of the Order Promising server or the Order Promising connector are running. The file locking built into the Order Promising datastore allows only one program to have access to Order Promising data at a time.

Starting the Server in Windows

To start the server in Windows:

From the Start menu, select Programs, EnterpriseOne Supply Chain Planning Order Promising 8.12.1, Order Promising Server.

Starting the Server in UNIX

To start the server in UNIX:

1. Log in as the root user.
2. Enter the following command to change directories:

   `cd` *`path`*`/scp/8.12.1/op/bin`

   where *path* is the directory path for Order Promising.

3. Enter the following command:

   ```
   ./run_opserver.sh
   ```

   The server status is displayed in the terminal window.

Signing In To EnterpriseOne Order Promising

To log in to EnterpriseOne Order Promising:

1. Open a web browser.
2. Enter the following URL in the Address field:

   `http://hostname:port_number/context_name`

   where *hostname*is the host name or IP address of the host, *port_number* is the port number on the host, and *context_name* is the name you have assigned to EnterpriseOne Order Promising.

   For example, `http://localhost:9080/Order Promising Web.`
3. Sign in using your name as the user ID.

Chapter 4

Defining Service Objectives

This chapter provides an overview of service objectives and discusses how to:

- Configure service objectives
- Configure fulfillment rules
- Compare fulfillment rules

Note. This chapter is required. You must complete the tasks discussed in this chapter to define service objectives that govern order promises made in Order Promising.

Understanding Service Objectives

This section discusses:

- Logistics rules
- Manufacturing rules
- Sourcing rules
- Delivery rules
- Product substitution rules
- Promising preferences
- Ranking
- Default service objective

Service objectives represent important priorities in your business model. Using sets of rules that apply to the sales orders of a particular customer, you can define service objectives and then use the service objectives as the basis for fulfilling sales orders. Alternatively, you can override the customer service objective by assigning a different service objective to a sales order.

Service objectives are made up of a series of fulfillment rules that govern the details of the order promise. Fulfillment rules can be created to govern the logistics, manufacturing, delivery, product substitution, and sourcing of a sales order. For example, you can use service objectives to specify whether product substitution can occur for a particular line item. Assigning rules to all customers, specific customers, or groups of customers allows you to condense the full sourcing matrix. When you define rules, you set various combinations of rules, from the most specific to the most general.

Order Promising enables you to evaluate different service objectives to determine which rules best suit your business priorities. Service objective rules are typically defined once during implementation and used to influence how the system promises a sales order. You can evaluate and modify rules after their initial creation and add additional rules to support changes in your business strategy.

Service objectives are associated with specific customers or sales orders in EnterpriseOne. Any service objectives that are created in Order Promising must be defined in the EnterpriseOne 34A/BO UDC table. A default service objective called "Standard" is provided by both EnterpriseOne and Order Promising. It can be modified in Order Promising if required.

Logistics Rules

Defining logistics rules enables you to manage the internal distribution processes of your enterprise according to the applicable service objective used when promising an order. You can open or close shipping lanes during a specific time period or horizon timefences.

Logistics rules can be tailored for:

- A specific customer, group of customers, or all customers.
- A specific item, item group, or all items.
- A specific lane.
- A specific transport mode.

Manufacturing Rules

Manufacturing rules allow you to leverage the power of Order Promising Capable to Promise (CTP) functionality to manufacture items that are not currently available in inventory. With the manufacturing rules, you can make routings available or unavailable for a specific time period or horizon timefence to reflect constrained capacities or processes unavailable due to maintenance. In addition, available routings can be set up to use premium capacity and premium material.

Manufacturing rules can be tailored for:

- A specific customer, group of customers, or all customers.
- A specific item, item group, or all items.
- A specific location, location group, or all locations.
- A specific routing.

Sourcing Rules

Creating sourcing rules allows you to influence the sourcing method for specific items and for different customers or customer groups. Order Promising allows you to source an item by proximity searching (locating the closest geographic locations) or by preferred sourcing.

When you enable proximity searching, Order Promising can automatically determine the closest location to the shipping address on the customer order by using a global database of 2.85 million cities, towns, and villages identified by their geographic coordinates. When an order is entered for a specific customer, the system initiates a proximity search using the city longitude and latitude. Based on these geographic coordinates, Order Promising can calculate the distance, delivery cost per unit ordered, and lead time.

If you choose to source items using proximity sourcing, you can return up to five suggested locations sorted by your choice of delivery cost, distance, or lead time.

Note. For the system to perform an effective proximity search, addresses must be correctly specified in sales orders.

Alternatively, you can specify up to four preferred sources to search for available items or item groups.

Sourcing rules can be tailored for:

- A specific customer, customer group, or all customers.
- A specific item, item group, or all items.

Delivery Rules

Delivery rules help Order Promising determine the final availability date and delivery cost to the customer. You can use delivery rules to estimate transportation lead times and delivery costs from the final shipping point to the customer. Lead time can be calculated as a combination of fixed lead time and variable lead time based on distance. For example, you can specify a fixed lead time of five days with an additional day of lead time added for every 500 miles of distance travelled. Delivery costs are calculated by weight.

Delivery rules can be tailored for:

- A specific location, location group, or all locations.
- A specific country, or all countries.
- A specific state, or all states.
- A specific city, or all cities.

Product Substitution Rules

Creating product substitution rules allows you to specify the items to substitute when the customer's primary choice of item is unavailable. When specifying the substitution item or group, you can also indicate the substitution ratio, cost, and multiple. Specific substitutions can be made unavailable.

Order Promising can perform the following product substitutions:

- Item to item.
- Item to group.
- Group to group.
- Group to item.

You can define multiple item substitutes for a specific item by creating multiple substitution rules for that item. When you create a substitution rule that contains a substitute item, you can assign a priority to the substitute item that the system will use to determine a substitution hierarchy.

Product substitution rules can be tailored for:

- A specific customer, group, or all customers.
- A specific item or item group.
- A substitution item or group.

Promising Preferences

Specifying promising preferences involves the selection and ordering of solution types. Solution types are preferences that the server uses to determine fulfillment options for an order when the standard fulfillment option, which tries to fulfill the order from available inventory, does not return a solution. You can select from the following solution types:

- Product Substitution
- Multi-Sourcing
- Manufacturing
- Premium Manufacturing
- Upstream Allocation

You select solution types in the order that you want the system to search when considering fulfillment options. If you do not select a solution type, the system will not consider any rules set for the specific solution type. For example, if you do not choose manufacturing as a solution type, the system will not consider any manufacturing fulfillment rules that you define. As a result, in this case, Order Promising will only use inventory that is available to promise (ATP) and will not use capable to promise (CTP) functionality.

Note. Configured items only support a Manufacturing preference type.

Ranking

The Order Promising server performs searches for rules in a sequence from the most specific to the most general, as follows:

- Rules that are customer specific
- Rules that apply to groups of customers
- Rules that apply to everyone

You must select ranking factors to determine the order in which the server returns promising solutions. This process provides maximum flexibility in configuring your business priorities. It is important to select appropriate ranking factors when you define a service objective in order to return promising solutions that meet the business priorities of your enterprise.

The following fields define the criteria used to fulfill orders:

Value	Use the value of the sales order as a priority for ranking promising solutions.
Delivery Cost	Use the total cost of delivering the order from the final shipping point to the customer as a priority for ranking promising solutions.
Substitutions	Use the total number of line items substituted on the order as a priority for ranking promising solutions.
Back Orders	Use the total number of backordered line items as a priority for ranking promising solutions.
Margins	Use the profit margin associated with the fulfillment of the order as a priority for ranking promising solutions. The profit margin is calculated as: (Price– Cost) / Price) x 100.
Profit	Use the amount of profit associated with the fulfillment of the order as a priority for ranking promising solutions. The profit is calculated as: Price– Cost.
Cost	Use the total cost of fulfilling the sales order as a priority for ranking promising solutions. The cost does not include any opportunity costs of shipping the order late. This value represents the item cost and any distribution costs, manufacturing (materials and resource time), and final delivery cost of each line item on the sales order.

Default Service Objective

A default service objective is provided by both EnterpriseOne and Order Promising called *Standard* . It is the service objective that is used for any orders from customers not associated with a specific service objective in EnterpriseOne. If not modified, it will fulfill orders through ATP only. It is recommended that you modify this default service objective to reflect your company's fulfillment preferences for sales orders that do not have associated service objectives.

Configuring Service Objectives

This section discusses how to:

- Create service objectives.
- Modify service objectives.
- Delete service objectives.
- Activate service objectives.
- Deactivate service objectives.
- Duplicate service objectives.

Pages Used to Configure Service Objectives

Window Name	*Navigation*	*Usage*
Add a Service Objective	Home Page, Service Objectives Click Add.	Create service objectives.
Edit a Service Objective	Home Page, Service Objectives Select a service objective. Click Edit.	Modify service objectives.

Creating Service Objectives

Access the Add a Service Objective page.

1. Complete the following fields:

Name Specify a name for the service objective.

Description Specify a brief description for the service objective that allows you to quickly identify the scope of the service objective. This field is optional.

Maximum Queries The maximum number of proximity sourcing locations to be considered when promising with this service objective. Up to four sourcing locations can be specified. The greater the number of sourcing locations specified, the more likely that items will be found to fulfill order requirements. The sourcing specified in this field applies only to available-to-promise inventory. Capable-to-promise sourcing is set in a sourcing fulfillment rule in the proximity sourcing field.

2. Specify promising preferences in the following fields:

Available Solution Types Select the solution types for the server to use to fulfill the order if it can't be fulfilled from available inventory. Valid values are: *Product Substitution, Multi-Sourcing,Manufacturing,Premium Manufacturing,* and *Upstream Allocation.* If no solution types are selected, the system will not consider fulfilling the order using Order Promising's capable-to-promise functionality.

Solution Types To Consider This field contains an ordered list of the solution types that you select from the Available Solution Types list.

3. Specify ranking behavior in the following fields:

Available Factors	Select the ranking factors and the order of ranking for the server to use when filtering promising solutions. Ranking factors refine the order promise and allow you to generate fulfillment results that best meet your service objectives. You can choose up to a maximum of four factors. Ranking factors eliminate as many of the partial solutions as possible.
Factors To Use	This field contains the ranking factors that you select from the Available Factors list.

4. Click Save.

Modifying Service Objectives

Service objectives can be changed to reflect your changing business requirements.

Access the Edit Service Objective page.

1. Select the check box next to the service objective you want to edit.
2. Click the Edit button.
3. Modify the values according to your business requirements.
4. Click Save.

Note. The *Standard* service objective should be modified to reflect your fulfillment preference for orders not assigned a specific service objective.

Deleting Service Objectives

Service objectives can be permanently deleted if they are no longer required. Alternatively, if you don't want to use a service objective temporarily, you can deactivate it.

Access the Service Objectives page.

1. Select the check box next to the service objective you want to delete.

   Multiple service objectives can be selected.

2. Click Delete.

Activating Service Objectives

Service objectives must be activated to be used during promising.

Access the Service Objectives page.

1. Select the check box next to the service objective you want to activate.

   Multiple service objectives can be selected.

2. Click Activate.

Deactivating Service Objectives

Service objectives can be deactivated to enable you to make changes. While a service objective is deactivated, it is not used during promising.

Access the Service Objectives page.

1. Select the check box beside the service objective you want to deactivate.

   Multiple service objectives can be selected.

2. Click Deactivate.

Duplicating Service Objectives

You can base a new service objective on an existing service objective by making a duplicate of the original. The duplicate can then be edited to meet your requirements.

Access the Service Objectives page.

1. Select the check box beside the service objective you want to duplicate.
2. Click Duplicate.

Configuring Fulfillment Rules

This section discusses how to:

- Define logistics rules.
- Define manufacturing rules.
- Define sourcing rules.
- Define delivery rules.
- Define product substitution rules.
- Edit fulfillment rules.
- Duplicate fulfillment rules.
- Change rule priority.
- Delete fulfillment rules.

Pages Used to Configure Fulfillment Rules

Window Name	*Navigation*	*Usage*
Add a Logistics Rule	Home Page, Service Objectives Select a service objective by clicking on the link. Click in the Logistics Rules grid. Click Add.	Define logistics rules.
Add a Manufacturing Rule	Home Page, Service Objectives Select a service objective by clicking on the link. Click in the Manufacturing Rules grid. Click Add.	Define manufacturing rules.
Add a Sourcing Rule	Home Page, Service Objectives Select a service objective by clicking on the link. Sourcing Rules grid. Click Add.	Define sourcing rules.
Add a Delivery Rule	Home Page, Service Objectives Select a service objective by clicking on the link. Delivery Rules grid. Click Add.	Define delivery rules.
Add a Product Substitution Rule	Home Page, Service Objectives Select a service objective by clicking on the link. Product Substitution grid. Click Add.	Define product substitution rules.
Edit a Fulfillment Rule	Home Page, Service Objectives Select a service objective. Select a service objective by clicking on the link. Select the rule you want to edit. Click Edit.	Edit the rule to reflect your business requirements.

Window Name	*Navigation*	*Usage*
View Service Objective	Home Page, Service Objectives Select a service objective by clicking on the link.	Delete fulfillment rules Change fulfillment rule priority Duplicate fulfillment rules

Defining Logistics Rules

Access the Add a Logistics Rule page.

Customer Code or Group Select the customer code or customer group to which the shipping rule applies. Select an asterisk if the rule applies to all customers.

Item Code or Group Select the item code or item group to which the shipping rule applies. Select an asterisk if the rule applies to all items.

Lane Code Select the transport lane to use to for shipping items.

Transport Mode Select the type of transportation that is used to service the specified lane. Select an asterisk to specify all transport modes. The Transport Mode depends on the value that you specify for the Lane Code.

Open lane during promising Select this option to open the transport lane during the promising of a sales order.

Close lane during promising Select this option to close the transport lane during the promising of a sales order.

Specify dates Select this option to indicate the effective start date and end date for this shipping rule. The date must be entered in yyyy-mm-dd format.

Specify Timefence Select this option to indicate the number of days from the beginning of the horizon timefence that the effective period starts and ends for this shipping rule. Specify the following values:

- In the Start Timefence field, specify the number of days from the beginning of the promising horizon to the date that the rule becomes effective. Enter *–1* in this field to indicate that the rule starts immediately.
- In the End Timefence field, specify the number of days from the beginning of the promising horizon to the date that the rule is no longer valid. This value must be greater than the value in the Start Timefence field. Enter *–1* in this field to indicate the absence of an end date.

Click Save.

Defining Manufacturing Rules

Access the Add a Manufacturing Rule page.

Customer Code or Group Select the customer code or customer group to which the manufacturing rule applies. Select an asterisk if the rule applies to all customers.

Item Code or Group Select the item code or item group to which the manufacturing rule applies. Select an asterisk if the rule applies to all items.

Location Code or Group Select the location code or location group to which the manufacturing rule applies. Select an asterisk if the rule applies to all locations.

Routing Code Select the routing code to which the manufacturing rules applies.

Make the routing UNAVAILABLE Select this option to indicate that the specified routing is unavailable for this rule.

Make the routing AVAILABLE Select this option to enable the specific routing for this rule. As a result of enabling the routing, the specific manufacturing processes are considered for fulfillment options that require CTP. Optionally, you can select one of the following options:

- Select With Premium Capacity to consider premium capacity during a sales order inquiry. Premium Capacity is defined in the premiumCapacity attribute in the ResourceCapacity object.
- Select With Premium Material to consider premium materials during a sales order inquiry. Premium Material is defined in the premiumCost attribute in the InventoryPolicyPurchase object.

Specify dates Select this option to indicate the effective start date and end date for this manufacturing rule. The date must be entered in yyyy-mm-dd format.

Specify Timefence Select this option to indicate the number of days from the beginning of the horizon timefence that the effective period starts and ends for this manufacturing rule. Specify the following values:

- In the Start Timefence field, specify the number of days from the beginning of the promising horizon to the date that the rule becomes effective. Enter *–1* in this field to indicate that the rule starts immediately.
- In the End Timefence field, specify the number of days from the beginning of the promising horizon to the date that the rule is no longer valid. This value must be greater than the value in the Start Timefence field. Enter *–1* in this field to indicate the absence of an end date.

Click Save.

Defining Sourcing Rules

Access the Add a Sourcing Rule page.

Customer Code or Group Select the customer code or customer group to which the sourcing rule applies. Select an asterisk if the rule applies to all customers.

Item Code or Group Select the item code or item group to which the sourcing rule applies. Select an asterisk if the rule applies to all items.

Search for sourcing locations by proximity to the customer Select this option to source items using a proximity search. Specify the following values:

- In the Number of Locations field, select the number of locations that Order Promising searches for inventory availability during a sales order inquiry. You can choose up to five locations to be sourced by proximity to the customer, resulting in a higher likelihood that the specified items will be located in inventory. The higher the number of locations chosen for sourcing, the more time it may take to promise orders.
- In the Sort by field, you can sort the results of the proximity searches by lead time, shortest distance, or least delivery cost to determine the sourcing solution that best meets both the customer requirements and your company's sourcing preferences. The default is *sort by shortest lead time.*

Use preferred sourcing locations Select this option if you want to select the preferred locations from which the order is sourced. In the Location 1 - Location 4 fields, select the first, second, third, and fourth preferred locations from which to source items. You must specify locations if a proximity search is disabled. If a proximity search is enabled, EnterpriseOne Order Promising ignores these fields. The more locations specified, the greater likelihood that the items being sourced will be located. The higher the number of locations chosen for sourcing, the more time it may take to promise orders.

Click Save.

Defining Delivery Rules

Access the Add a Delivery Rule page.

Source Location Code or Group Select the location code or location group from which a shipment originates. Select an asterisk if the delivery rule applies to all locations.

Destination Country Enter the country to which the delivery rule applies. Enter an asterisk if the delivery rules applies to all countries.

Destination State Specify the state to which the delivery rule applies. Valid values are * (any state/province) or the name of a specific state/province.

Destination City Specify the city to which the order is shipped. Valid values are * (any city) or the name of a specific city.

Fixed for all destinations Select this option if you only want to use a fixed lead time as the main factor when determining a delivery strategy. Specify the amount of time in days necessary to ship orders from the source location to the destination.

Variable by the distance to the destination Select this option if you want to calculate the lead time based on a fixed lead time and the shipping distance. Order Promising combines both the fixed and variable components to determine the lead time. For example, if the fixed component is five days and the variable component is 500 miles, Order Promising calculates the total lead time to be five fixed days plus one additional day of lead time for every 500 miles required to deliver the order.

Fixed Component — Specify the fixed number of days of lead time.

Variable Component — Estimate the distance that can be travelled in a day. The amount specified will allow Order Promising determine the variable lead time based on travel distance.

Delivery Cost Specify the delivery cost based on the unit of measure selected.

Click Save.

Defining Product Substitution Rules

Access the Add a Product Substitution Rule page.

Customer Code or Group Select or enter the customer code or customer group to which the product substitution rule applies. Select an asterisk if the rule applies to all customers.

Item Code or Group Select or enter the item code or item group to which the product substitution rule applies.

Substitution Item Code or Group Select or enter the substitution item code or group to be used if the originally specified item code or group is not available.

Make the substitution UNAVAILABLE Select this option to disable the specified item substitution.

Make the substitution AVAILABLE Select this option to enable this item substitution. If you select this option, you must define substitution parameters in the appropriate fields.

Substitution Ratio Select the quantity of substituted products or materials used for each unit of the original product or material. The default value is *0.0*.

Substitution Cost Specify the penalty cost charged to the customer per unit to make the substitution. This penalty cost is combined with the actual cost of the substitution item to determine the per unit cost. The default value is *0.0*.

Substitution Multiple Specify the shipping multiple for the substituted product. For instance, some items are only available in multiples of six. When insufficient stock is available to fulfill an order of the original product, the allocation of substitutions is based on this value. However, if the substitution multiple is greater than the planning multiple, Order Promising may not be able to properly allocate an order. The default value is *0.0*.

Substitution Preference Specify the substitution preference for the substitute item or group. Order Promising uses this field to create a substitution hierarchy if you have defined multiple production substitution rules for the same item or item group. Valid values are *1* to *100*. The default value is *1*.

Click Save.

Editing Fulfillment Rules

Access the Edit a Rule page.

1. Modify the values according to your business requirements.
2. Click Save.

Duplicating Fulfillment Rules

Order Promising allows you to duplicate current fulfillment rules, eliminating the reentry of key information. In many cases, you can have multiple fulfillment rules that are essentially the same except for small differences.

Access the View Service Objective page.

1. Select the check box beside the fulfillment rule you want to duplicate.
2. Click Duplicate.
3. Modify the values of the new rule.
4. Click Save.

Changing Rule Priority

A service objective may have numerous fulfillment rules in any given category. Order Promising enables you to prioritize the application of the rules when fulfilling an order. For example, if there are five substitution items for a given item, you can indicate the order of substitution.

Another general principle to be considered is that it is best to put the most specific rules at a higher priority than general rules. For example, a specific customer's delivery rule should be at a higher level than the delivery rules for all customers.

Access the View Service Objective page.

1. Select a service objective by clicking the link.
2. For the rule for which you want to change the priority, do one of the following:
   - Click Raise Priority to increase the priority of the rule
   - Click Lower Priority to decrease the priority of the rule.

Delete Fulfillment Rules

When fulfillment rules are no longer useful, you can delete them.

Access the View Service Objective page.

1. Select a service objective by clicking the link.
2. Select the rule you want to delete.
3. Click Delete.

Comparing Fulfillment Rules

This section discusses how to:

- Compare service objective fulfillment rules.
- View service objective rules.
- Duplicate fulfillment rules.
- Edit fulfillment rules.
- Delete fulfillment rules.

Pages Used to Compare Service Objectives

Window Name	***Navigation***	***Usage***
Service Objectives	Home Page, Service Objectives	Compare service objective rules
Compare Rules	Home Page, Service Objectives Click the service objective's link from the comparison table.	View, duplicate, edit, and delete fulfillment rules

Comparing Service Objective Fulfillment Rules

Order Promising enables you to compare the fulfillment rules set for different service objectives for a specific rule category, such as manufacturing. The fulfillment rules for the selected service objectives are displayed in a table for easy comparison, sorted in order of priority. From this table, fulfillment rules can be duplicated, deleted, or edited.

Access the Service Objectives page.

1. Select the service objectives whose fulfillment rules you want to compare.
2. Select a fulfillment rule category from the drop-down list box.
3. Click Go.

Duplicating Fulfillment Rules

You can duplicate fulfillment rules from the comparison results.

Note. Fulfillment rules can only be duplicated within the same service objective.

Access the Compare Rules page.

1. Select the fulfillment rule you want to duplicate.
2. Click Duplicate.

   The duplicate rule appears at the bottom of the comparison table with the same name. The rule can now be edited.

Editing Fulfillment Rules

After comparing your fulfillment rules in different service objectives, you can edit specific rules directly from the comparison table. This option gives you capability to modify an existing rule so that it is more like a rule from another service objective.

Access the Compare Rules page.

1. Select a fulfillment rule from the comparison table.
2. Click Edit to make changes to the fulfillment rule.
3. Click Save.

Deleting Fulfillment Rules

The comparison table enables you to view all the fulfillment rules related to a specific rule category. Any extraneous fulfillment rules can be deleted directly from the comparison table.

Access the Compare Rules page.

1. Select the fulfillment option you want to delete.
2. Click Delete.

   The fulfillment option is removed from the comparison table.

Chapter 5

Simulating Sales Order Promising

This chapter provides an overview of simulated order promising and describes how to:

- Manage sales orders.
- Promise sales orders.

Understanding Simulated Order Promising

Order Promising uses a sophisticated algorithm to promise the fulfillment of sales orders that are created in EnterpriseOne Sales Order Management or other integrated systems. Order Promising maintains and stores a representation of the data within your supply chain. Throughout the Order Promising cycle, this representation is updated to reflect the changing demands and constraints that affect your supply chain. Order Promising uses this information to allocate available inventory and capacity in an enterprise to fulfill a sales order inquiry.

To determine whether your service objectives are providing the results you want, you can simulate the promising of orders before connecting with EnterpriseOne. Within Order Promising, you can create, edit, duplicate, and delete a set of prototype sales orders that you can use to test the functioning of your service objectives and the configuration of the Order Promising server. Simulated sales orders support the full range of options available in EnterpriseOne sales orders including allowing partial order shipments, backorders, partial line shipments, multisourcing, and substitutions to ensure that the simulated promising results are realistic.

After promising a simulated sales orders against the current Order Promising model, the Promising Results page displays detailed information about how the order can be fulfilled by the requested date including the quantities available from inventory, the quantities that can be manufactured, the number of items on backorder, and number of substitutions, the order fill rate, cost, price, delivery cost, profit, and profit margin. In addition, the Detail Results area provides fulfillment information about each line item on the sales order including quantities available, dates, source, prices, and profit margins. If you are not satisfied with the results, you can change the simulated sales order and repromise the order.

Note. The promising of simulated sales orders does not affect the availability of inventory or resources for real-time sales orders because simulated sales orders cannot be committed. Both simulated and real-time sales orders can be run simultaneously if desired.

See Also

"Defining Service Objectives"

Managing Simulated Sales Orders

This section discusses how to:

- Create simulated sales orders
- Edit simulated sales orders
- Duplicate simulated sales orders
- Delete simulated sales orders

Understanding Simulated Sales Orders

Sales orders can be created to simulate orders from EnterpriseOne. Designed to emulate the EnterpriseOne Sales Order Entry form, sales orders are comprised of a header section and detail line section. The header section contains information such as the sales order number, customer code, customer address, service objective, and fulfillment preferences. The sales order detail section contains specific information about the item requested, the quantity, and the requested date.

Sales orders can be edited to improve their profitability or fulfillment. You can base a new sales order on an existing sales order by using the duplication feature. When no longer required, sales orders can be deleted.

Pages Used to Manage Simulated Sales Orders

Window Name	***Navigation***	***Usage***
Simulated Sales Orders	Home, Sales Orders	Create, edit, duplicate, delete or view a simulated sales order.
View Sales Order	Home, Sales Orders Select a simulated sales order by clicking on the link.	View the header and details of a simulated sales order.
Add Sales Order Header	Home, Sales Orders Click Add.	Add a simulated sales order header.
Add Sales Order Detail	Home, Sales Orders Select a simulated sales order by clicking on the link. Sales Order Details grid. Click Add.	Add detail lines to a simulated sales order.

Window Name	*Navigation*	*Usage*
Edit Sales Order Header	Home, Sales Orders Select a simulated sales order by clicking on the link. Click Edit.	Edit a simulated sales order's header.
Edit Sales Order Detail	Home, Sales Orders Select a simulated sales order by clicking on the link. Sales Order Details grid. Select the detail line you want to edit. Click Edit.	Edit a simulated sales order detail line.

Creating Simulated Sales Orders

This section discusses how to:

- Create a sales order header.
- Create sales order detail lines.

Creating a Sales Order Header

Access the Add Sales Order Header page.

Sales Order Code Specify the sales order code. You can enter any combination of letters or numbers.

Customer Code or Group Select the customer code to be used for this sales order.

Country Specify the country where the order is to be shipped.

State Specify the province or state where the order is to be shipped.

City Specify the city where the order is to be shipped.

Service Objective Select a service objective that Order Promising will use when promising the sales order. Only activated service objectives are available.

Allow Partial Order Shipments Select this option to allow portions of an order to be shipped.

Allow Backorders Select this option to promise items that are currently out of stock when they become available. If this option is not enabled, only those line items that are available on the request date are shipped.

Allow Partial Line Shipments	Select this option to allow the shipment of individual line items as they become available. This option can only be selected when the Allow Backorders and Allow Partial Order Shipment are also selected.
Allow Multisourcing	Select this option to allow items to be shipped from, or manufactured at, multiple locations.
Allow Product Substitution	Select this option to enable product substitution when insufficient quantities of the preferred item are not available.

Click Save.

Creating Sales Order Detail Lines

After the sales order header has been created, you can add the sales order detail lines.

Access the Add Sales Order Detail page.

Item Code	Select the item code for the item being ordered.
Quantity	Specify the quantity of the item being ordered.
Planning Unit	Specify the standard planning unit of the item being ordered.
Planning Multiple	Specify the planning multiple of the item being ordered. For example, a planning multiple of 12 indicates that items are included in the order only in groups of 12.
Planning Unit Price	Specify the price of the of the item being ordered in the planning unit of measure.
Request Date	Specify the date when the customer wants to receive the item. Use the format yyyy-mm-dd.

Click Save.

Editing a Simulated Sales Order

This section discusses how to:

- Edit a sales order header.
- Edit sales order line details.
- Delete sales order line details.

Editing a Sales Order Header

Access the Edit Sales Order Header page.

1. Make changes to the sales order header.

2. Click Save.

Editing Sales Order Line Details

Access the Edit Sales Order Detail page.

1. Make changes to the line items.
2. Click Save.

Deleting Sales Order Line Details

Access the View Sales Order page.

1. Select the sales order line items you want to delete.
2. Click Delete.

Duplicating Simulated Sales Orders

Access the Simulated Sales Orders page.

1. Select one or more simulated sales orders that you want to duplicate.
2. Click Duplicate.

   The text "Copy of" is appended to the original name of the sales order. To change the name, edit the sales order header.

Deleting Simulated Sales Orders

Access the Simulated Sales Orders page.

1. Select the simulated sales orders you want to delete.
2. Click Delete.

Promising Sales Orders

This section discusses how to:

- Promise a sales order
- Evaluate promising results

Understanding Promise Results

When a sales order is promised, Order Promising fulfills the order based on first the available to promise (ATP) timefence and then the service objective set for the order. The ATP timefence is a user-defined period of time set at the beginning of the promising horizon that helps Order Promising determine whether to fulfill an item order from available inventory or try to manufacture it based on the customer's request date. The service objectives include manufacturing, shipping logistics, delivery, sourcing, and product substitution information that Order Promising uses to determine the costs associated with the order, and the order profitability.

If you are not satisfied with the promise, return to the sales order, modify it, and then repromise the sales order. Alternatively, you can make changes to the service objectives, and then repromise the sales order to see if your results have improved.

Note. Although a promise can be generated for a simulated sales order, the promise cannot be committed, so inventory and resources are never reduced. Only sales orders created in EnterpriseOne that are transmitted to Order Promising can be promised and committed.

See Also

"Defining an ATP Timefence"

"Defining Service Objectives"

Page Used to Promise Sales Orders

Window Name	*Navigation*	*Usage*
Promising Results	Home, Sales Orders Select a sales order by clicking on the link. Click Promise.	View promising results for a sales order.

Promising a Sales Order

To promise a sales order:

1. Select a sales order by clicking the appropriate sales order in the sales order list.
2. Click Promise.

   The promising results page appears.

Evaluating Promising Results

Access the Promising Results page.

1. View the following fields in Header Results:

Service Objective	The service objective that you selected prior to promising this sales order.
Number of ATP Items	The number of line items in this sales order that have been promised using available inventory.
Number of CTP Items	The number of line items in this sales order that have been promised using manufacturing capacity.
Number of Backorders	The number of items in the sales order that are unavailable on the request date. These items will be shipped when they become available.
Number of Substitutions	The number of product substitutions made if line items in this sales order are unavailable on the shipping date.
Order Fill Rate	The percentage of the sales order that was allocated using either current inventory or manufacturing capacity. The order fill rate is the total of the line fill rate values.
Order Cost	The total cost of all of the items associated with this sales order.
Order Price	The total price of all of the items associated with this sales order.
Order Delivery Cost	The total costs involved with delivering the items in this sales order to the customer.
Order Profit	The total profit realized from the sale of all items in the sales order.
Order Margin	The profit margin associated with this sales order.

2. View the fields in the promising details.

Line	The line number on the sales order associated with the line item.
Requested Item	The item ordered by the customer.
Available Item	The item that is available for the customer. This item can be the requested item or a substitute item.
Available Amount	The quantity of this line item that is available on the shipping date.
Requested Date	The date that the customer wants this line item to be delivered and available at their location.
Available Date	The date that this line item can be available at the customer location.
Ship Date	The date that this line item ships from its final distribution point to the customer.
Pick Date	The date that this line item is prepared for shipment to the customer.
Ship Location	The location from which this line item is shipped to the customer.
Price	The total extended price of this line item.
Profit	The projected profit generated by this line item.
Margin	The profit margin associated with this line item.
Line Fill Rate	The percentage of the line item quantity that can be fulfilled.
Suspected Cause	The suspected reason why the system could not fulfill the requested line item.

3. Click View Sales Order to review and modify the sales order, if desired.

See Also

Appendix D: Understanding Sales Order Inquiry Error Codes

Chapter 6

Searching for Available Inventory

This chapter provides an overview of available inventory and discusses how to search for available inventory.

Understanding Available Inventory

Order Promising fulfills orders based on the availability of items in inventory, or your capability to manufacture them. Depending on how you have configured Order Promising, Order Promising might attempt to fulfill the order from available inventory, substitute an item, ship a partial order, or manufacture all or part of the order.

At any time, you can view the availability of an item on a specific date in a two-week grid, based on the promising horizon. You can check item inventory before processing a simulated sales order and use inventory information to analyze your promise results.

Note. Because Order Promising is regularly confirming new orders, the available inventory is constantly changing. Inventory results are only accurate at the moment when they are queried.

Searching for Available Inventory

This section lists the page used to search for available inventory and discusses how to search for available inventory:

Page Used to Search for Available Inventory

Window Name	*Navigation*	*Usage*
Available Inventory	Home, Inventory	Review available inventory on a specific day.

Searching for Available Inventory

Access the Available Inventory page.

1. Complete the following fields:

   Item Code Specify the item code for the item in inventory.

   Location Code Select the location from which you want to source the item. Select an asterisk to search all locations.

2. Click Search.

   A two-week grid displays the daily inventory levels of the specified item at the locations requested. The results displayed are based on the default unit of measure.

3. Click Next 2 weeks to move further into the promising horizon. You can also click Previous 2 weeks to return to a previous time period.

Chapter 7

Allocating Resources

This chapter provides an overview of allocations and discusses how to:

- Create allocation contracts.
- View allocation contracts.

Understanding Allocations

In the past, large enterprises were required to make significant capital investments in order to expand their manufacturing capacity. The significant expense incurred by capacity expansion created significant financial risk for the enterprise if the expected demand for their products did not materialize. If demand was greater than the capacity of the enterprise, customers were exposed to higher prices. In today's competitive manufacturing environment, effective management of the capacity of your enterprise is critical in order to satisfy customer demand without building costly excess capacity. Increasing demand volatility in the manufacturing sector presents an issue for large corporations.

The availability of a precise amount of capacity at specific intervals in the planning horizon is a crucial factor in efficient supply chain operation. In today's competitive business environment, your largest customers are strategically important in your business plan. The ability to guarantee your customers a large order or the capacity to build a large order by a certain date is now a necessity. Order Promising provides you with allocation management functionality that allows you to allocate items and resource capacity into the future for specific customers and sales channels. With Order Promising, you can negotiate higher margins with your customers using allocation contracts that guarantee a future supply to customers while leveraging your existing capacity. Order Promising allows you to sell items and resources to customers in the future using contracts, enabling your customers to minimize the financial risks associated with demand volatility. Priority customers will appreciate the ability to allocate capacity for important orders.

Allocation management also allows you to minimize costly capital investments traditionally associated with capacity expansion. When customers reserve capacity ahead of time, capacity reservations provide you with the ability to see the future demand of your most important customers, allowing you to scale capacity in line with demand and reduce excess capacity. Allowing your customers to allocate capacity reduces your exposure to demand volatility, and allows you to quickly react to changing market conditions. Allocation contracts enable you to negotiate higher margins with your customers in return for guaranteed capacity. Using Allocation contracts gives you the necessary lead time that is often required to perform facility or machine configuration or to set up outsourcing agreements for large orders. Unused capacity contained in expired reservations can be reused. You can further leverage the manufacturing capacity of your enterprise by allocating reserved capacity that has been unused to other customers.

Order Promising provides the capability to view current allocation contracts through the web application. These allocation contracts are taken into consideration by Order Promising when promising sales orders, and ensure your customers a steady flow of key items.

Creating Allocation Contracts

This chapter provides an overview of allocation contracts and discusses how to create an allocation contract.

Understanding Allocation Contracts

Item allocations are administered in Order Promising through the use of contracts. For each item to be allocated for a customer at a specific location, a contract needs to be created. The contract contains information about the customer, the item to be allocated, the resources associated with the item and their location, the quantity of the item to be allocated on a weekly basis, and when the allocations start. Allocation contracts also specify the expiry timefence, the number of weeks from the beginning of the horizon after which weekly allocations expire, and the specific day of the week when the allocation expires. If the customer does not use their allocation by the expiry timefence, the resources are automatically released for use by other customers.

Creating an Allocation Contract

Allocation contracts are created in the ResourceAllocation.xml file located in the Datastore directory. The ResourceAllocation.xml file contains information about the main customer contract such as the item to be allocated, the start and end dates for the contract, and customer information, and the contract specifics. The items and resources specified in a contract must be contained in the Order Promising data model.

See Also

"Appendix C: Understanding the Order Promising XML Format" for more information about how understand the Order Promising XML format.

Viewing Allocation Contracts

This chapter discusses how to:

- View a list of allocation contracts.
- View allocation details for a contract.
- View weekly allocations for a resource.

Common Elements Used in This Section

Allocation ID A unique code that identifies the contract. When you create a new contract, Order Promising prompts you to specify an allocation ID. The allocation ID can be unique or it can correspond with another code used for the contract in an external system.

Customer Code The code for the customer involved in the contract.

Item Code The code for the item that you want to allocate. Usually, this is the finished item..

Item Description The description of the item you want to allocate.

Location Code The code of the location where the item you want to allocate is manufactured or stored.

Resource Code The code for the resource that is being allocated. Resources must be defined in the manufacturing data model before they can be allocated.

Resource Type The type of resource being allocated. Valid values are: *crew*,*item*,*machine*, and *tool*.

Unit of Measure The standard planning unit of the allocated item.

Pages Used to View Allocation Contracts

Window Name	*Navigation*	*Usage*
Allocation Manager View	Home, Allocation Manager	View a list of customer contracts.
View Allocation Detail	Home, Allocation Manager Select a customer contract by clicking on the link.	View the allocation details for a customer contract.
Weekly Allocation	Home, Allocation Manager Select a customer contract by clicking on the link. Select a resource by clicking on the link.	View the weekly allocations for a resource.

Viewing a List of Allocation Contracts

Access the Allocation Manager View page.

To view a list of allocation contracts, refer to the following fields in the Allocation List table:

Start Date	The start date of the contract.
End Date	The end date of the contract.

Viewing Allocation Details for a Contract

Access the View Allocation Detail page.

To view the detail lines for an allocation contract, refer to the following fields in Resource Allocation:

Use Reservation Only	Specifies whether or not the solver can use item or capacity outside of the allocated reservation.
Expiry Timefence	The number of weeks before the weekly allocation expires, starting from the horizon.
Expiry Day	The day of the week when allocations expire. This field works in conjunction with the number of weeks specified in the Expiry Timefence.Valid values are: *Monday,Tuesday*, *Wednesday,Thursday,Friday,Saturday*, and*Sunday.*

Viewing Weekly Allocations for a Resource

Access the Weekly Allocation page.

To view the weekly allocations for a resource, refer to the following fields in Weekly Allocation:

Start Date	The start date for the weekly allocation.
Allocated Capacity	The number of units of weekly capacity allocated for the item at the location specified.

Chapter 8

Administering EnterpriseOne Order Promising 8.12.1

This chapter provides an overview of Order Promising administration and discusses how to:

- Configure the promising server.
- Configure the datastore.
- Monitor the server.
- View system configuration.

Understanding EnterpriseOne Order Promising Administration

This section discusses the following administrative themes:

- Server configuration
- Default directory structure
- Datastore configuration
- Server monitoring
- Inventory status
- Promising queue
- Server logging
- Batch startup scripts
- Server executable files

Server Configuration

Administration of Order Promising is accomplished by modifying a series of configuration variables that control the behavior of the promising server, datastore, and gateway interfaces. Depending on the scope of your Order Promising implementation, you can modify configuration variables for the following purposes:

- Facilitate the deployment of web components

- Configure system logs
- Enable initial inventory and resource allocation reports

Default Directory Structure

The files that are shared by multiple components are installed in the `/scp/8.12.1/op` directory. The following table lists the directories that the installation program creates in the version directory:

Directory	***Contents***
`/data/`	The datastore directory that contains the data files used by EnterpriseOne Order Promising.
`/bin/`	The EnterpriseOne Order Promising executables that are specific to a particular platform.
`/cfg/`	The location of the server configuration file that determines the behavior of the promising server.
`/logs/`	The log files and reports that are created during the operation of EnterpriseOne Order Promising.
`/geo/`	The geographic database file that contains geographic coordinates used during a proximity search.
`/connector/`	The location of the data connector used for data transfer to and from EnterpriseOne Supply Chain Business Modeler.
`/jre/`	The location of the Java environment.
`/lib/`	The location of TCL libraries used by Order Promising.
`/Uninstall/`	The location of the uninstall application.
`/xsd/`	The location of the XML schema definitions used by the Order Promising datastore.

Datastore Configuration

The datastore is the central repository for enterprise data that arrives inbound from EnterpriseOne and the EnterpriseOne Order Promising data model. The datastore consists of an XML file structure that is updated dynamically when you make changes to the data model during enterprise data refresh or order promising. The XML architecture of the datastore facilitates a streamlined integration with external systems through EnterpriseOne Supply Chain Business Modeler.

You can configure the datastore to meet your promising requirements. The datastore configuration file is an XML file that you can edit in an XML editor and save to the appropriate directory. You must restart the EnterpriseOne Order Promising server for your datastore configuration changes to take effect.

The datastore has file locking to ensure that your model data isn't accidentally overwritten. At present, one session of either the Order Promising server or the Order Promising connector can update the datastore at a time.

Geographic Database

During the course of a proximity search, EnterpriseOne Order Promising retrieves a customer address and compares that address with Geographic Database entries. However, proximity searches can fail if the location information in the data model does not match the Geographic Database. For example, assume that a customer location is entered as Vancouver, BC, in the data model. However, in the Geographic Database, the customer location is listed as Vancouver, B.C. This results in a mismatch that causes the proximity search to fail.

Order Promising uses the GeographicAlias.xml file to reconcile customer location information in the data model with customer location information in the Geographic Database. The GeographicAlias.xml file is stored in the datastore in the *path*`/scp/8.12.1/op/data/opserver/datastore` directory. You can edit the GeographicAlias.xml file to include popular aliases for the following location parameters:

- The type of location (city, state or province, or country)
- The name of the location - for example, Hamilton
- A commonly used alias for the location - for example, CO to denote Colorado
- State or province and country filters

You can edit the GeographicAlias.xml file if a location does not exist in the Geographic Database, a spelling of the name of the location is different, or if you want to use abbreviations.

When you configure aliases for customer locations, you can add multiple aliases for each location. The United States can have the aliases US, U.S., USA, U.S.A., United States of America, and so on. Each alias must have its own entry in the GeographicAlias.xml file.

The following example illustrates a list of aliases used for a country in the GeographicAlias.xml file:

```
<countryName>United States</countryName>
<countryAliasList>
   <countryAlias>US</countryAlias>
   <countryAlias>USA</countryAlias>
   <countryAlias>US.</countryAlias>
   <countryAlias>USA.</countryAlias>
</countryAliasList>
```

The following example illustrates aliases for a state or province and cities in the GeographicAlias.xml file:

```
<stateProvince>
   <stateProvinceName>New York</stateProvinceName>
   <stateProvinceAliasList>
      <stateProvinceAlias>NY</stateProvinceAlias>
   </stateProvinceAliasList>
   <cityList>
     <city>
       <cityName>New York</cityName>
       <cityAliasList>
       <cityAlias>NY</cityAlias>
       <cityAlias>NYC</cityAlias>
       <cityAlias>New York City</cityAlias>
       </cityAliasList>
     </city>
  </cityList>
</stateProvince>
```

Customer Address Verification

EnterpriseOne Order Promising allows you to verify a customer address by finding the exact longitude and latitude coordinates for a customer's address by using the geolookup utility. The server uses the longitude and latitude coordinates when it conducts a proximity search. The geolookup utility is located in the *path* \scp\8.12.1\op\bin directory and can be initiated from a DOS or UNIX command prompt.

Datastore Recovery

Throughout the day, the in-memory Order Promising model is kept current with changes made in EnterpriseOne. Real-time messages are sent to Order Promising from EnterpriseOne representing:

- manual inventory adjustments
- sales orders
- procurement orders
- transfer orders

To maximize promising performance, the datastore is not updated with the real-time messages until the Order Promising server is properly stopped. In case of the failure of the Order Promising server, all incoming real-time messages from EnterpriseOne are recorded in the .requestJournal file located in the Datastore directory. The presence of this file ensures that all the promises made during the day are not lost if the server fails. In the event of a proper conclusion of an Order Promising server or connector session, two things happen:

- The real-time messages, which up to this point have only updated the in-memory Order Promising model, now update the datastore. After the datastore has been updated successfully, the contents of the .requestJournal file are cleared.
- The .dataStoreGuard file is removed, thereby allowing another session of either the Order Promising server or connector to begin.

However, in the event of the failure of either the Order Promising server or connector, the .dataStoreGuard file is not removed and the datastore remains locked. The contents of the .requestJournal.xml file remain intact, ready to be added to the datastore before the Order Promising server loads the model into memory.

See Also

"Recovering the Datastore if the System Fails"

Server Monitoring

EnterpriseOne Order Promising allows you to monitor vital system processes, view configuration details, and monitor the promising queue from the Administration page. You can use the Administration page to retrieve the following system information in real time:

- Promising statistics
- System errors
- Promising horizon settings
- Log files
- Server version and location
- Real time promising queue

The ability to monitor server processes is important in a production environment where multiple systems are integrated. Order Promising enables you to monitor server system processes, allowing you to evaluate server performance and perform system troubleshooting in the event that performance has decreased.

The Administration page displays real time system data that can be used to:

- Determine the length of the current promising session
- Determine server performance by viewing average promise time
- Monitor error counts for the promising process and data integrity

Inventory Status

When you start Order Promising, an ATP initialization file (named atpinit.txt) is written to the log directory. This file consists of two sections, each starting with a Horizon line. The Horizon line represents each day in the promising horizon.

The first section of the file contains inventory level data for all combinations items and locations throughout the promising horizon. Each line represents an item at a specific location. The first column contains the Location and Item codes. Each column that follows indicates the available inventory for that combination of item and location for each day in the promising horizon.

The second section of the file contains resource data for manufacturing. Resources at specific locations that are considered by the promising server are listed in this section. The first column contains the Location and Resource codes. Each column that follows indicates the resource availability for each combination of location and resource for each day in the promising horizon.

The ATP initialization file size is proportional to the size of your data model. In certain implementations in which a large data model is used, the file might take a significant amount of time to generate. You can disable the creation of the ATP initialization file using the atp-init option in the server configuration file.

See Also

"Configuring Server Variables"

Promising Queue

The EnterpriseOne Order Promising server promises sales orders on a first in first out (FIFO) basis. Incoming sales orders received from EnterpriseOne Sales Order Entry are queued in the order that they are received. Sales orders that contain a large number of line items take longer to promise than orders with fewer line items, which can cause the queue to grow as new sales orders are received. The number of sales orders in the promising queue depends on the size of the sales order that is being processed at any moment and the volume of sales orders that are incoming from Sales Order Entry.

The Administration page provides you with information about how many sales orders are waiting in the promising queue and how long they remain in the queue. Sales orders that are promised before the browser is refreshed will not appear in the promising queue. You may not see a promising queue at all if sales orders are relatively brief and the volume of incoming orders is manageable.

To compliment the promising queue, the Administration page displays a record of the order promises in the current session that have taken the longest to process. The "at a glance" availability of promising queue data and a record of the longest promises allows you to respond quickly to customers and ensures a faster resolution to any escalation that may occur.

Server Logging

EnterpriseOne Order Promising maintains a detailed server log file that captures all received messages and message responses. Messages that are recorded in the server log file are timestamped and the corresponding process identifier (PID) for each message is displayed. The server log file stores essential application information that can be useful if you need to perform troubleshooting tasks on the system.

The server log file is stored in the directory that you define during the configuration of the promising server. The default directory used is the `/scp/8.12.1/op/logs` directory that is created when you install Order Promising. This directory path can be modified at any time in the opserverConfig.xml file to output the server log file to an alternate location.

Note. To manage the size of the Order Promising server log file, the system administrator should copy and delete the file on a regular basis.

See Also

"Configuring the Promising Server"

Sales Order Allocation Exception Reporting

When a sales order is promised, Order Promising reviews the current data model to determine whether the order can be fulfilled by the date required by the customer. When committed, the required date is persisted by the EnterpriseOne Sales Order Management system, not by Order Promising. Between the date when Order Promising was queried and the customer's requested date, many unforeseen things can happen that might interfere with the timely fulfillment of the order. For example, manufacturing capacity at a specific plant may decrease or temporarily cease, affecting the production of the ordered items. Upon startup of the Order Promising server, Order Promising scans the data model provided by EnterpriseOne for sales orders that are in danger of not being fulfilled on time. These sales orders are listed in the Sales Order Allocation Exception Report, located in the `/scp/8.12.1/op/logs` directory.

Batch Startup Scripts

EnterpriseOne Order Promising includes startup scripts in the `/scp/8.12.1/op/bin/` directory that can be used to start and shut down the following components:

- EnterpriseOne Order Promising server
- Data connector for EnterpriseOne Supply Chain Business Modeler

The following scripts are provided:

Script	***Description***
run_opserver.bat	Starts the EnterpriseOne Order Promising server in batch mode.
stop_opserver.bat	Stops the EnterpriseOne Order Promising server in batch mode.
run_opconnector.bat	Starts the data connector that is used to retrieve enterprise data from EnterpriseOne Supply Chain Business Modeler as a part of a batch integration.

Server Executable Files

The executable files used by EnterpriseOne Order Promising are stored in the `/scp/8.12.1/op/bin` directory. When you install the software, the following files are saved to disk:

Executable File	***Description***
geolookup	A utility that allows you to verify the integrity of your customer addresses in the Geographic Database.

Executable File	*Description*
opserver	The promising server, the heart of EnterpriseOne Order Promising, maintains an in-memory representation of the state of the enterprise. When the server performs queries, it examines the order inquiry and attempts to allocate available inventory (ATP) or capacity (CTP) in an enterprise.
OpConnector	The data connector that is used to refresh the enterprise data stored in the datastore. Enterprise data is stored in EnterpriseOne Supply Chain Business Modeler and loaded into the datastore using the data connector executable and the refresh command.
stopserver	An executable file that stops the EnterpriseOne Order Promising server.
license	An executable file that starts the License Manager.

Configuring the Promising Server

This section discusses how to:

- Navigate server directories.
- Configure server variables.
- Define an ATP timefence for specific items.
- Prioritize the fulfillment of sales orders upon startup.

Navigating Server Directories

Server files for Order Promising are installed in the `/scp/8.12.1/op` directory. The following table lists the directories that the installation program creates in the version directory:

Directory	*Contents*
`/data/`	The datastore directory that contains the data files used by Order Promising.
`/bin/`	The Order Promising executables that are specific to a particular platform.
`/cfg/`	The location of the server configuration file that determines the behavior of the promising server.
`/logs/`	The log files and reports that are created during the operation of Order Promising.

Directory	*Contents*
/geo/	The geographic database file that contains geographic coordinates used during a proximity search.
/connector/	The location of the data connector used for data transfer to and from EnterpriseOne Supply Chain Business Modeler.
/jre/	The location of the Java environment.
/lib/	The location of TCL libraries used by EnterpriseOne Order Promising.
/Uninstall/	The location of the uninstall application.
/xsd/	The location of the XML schema definitions used by the EnterpriseOne Order Promising datastore.

Configuring Server Variables

Access the opserverConfig.xml file.

To configure server variables:

port Specify the port that the server will listen on. This port must be recognized by the webserver when deploying web components. The default port is 39000.

datastoreDir Specify the datastore directory where the data model is stored. If not specified, the default is the /data/opserver/datastore directory.

Note. You can set up a test directory by copying the contents of the /data directory into a different directory, and then refer to the new directory in the datastoreDir server variable.

schemaDir Specify the location of the repository schema file. If not specified, the default is the /xsd directory.

logFile Specify the location to save the server log file. If no value is specified, the default is the /logs directory.

logLevel Specify the level of detail contained in the server logs. Valid values are 0: errors only, 1: errors and warnings, 2: errors, warnings and informational messages, and 3: log everything, including the content of all incoming and outgoing messages.

geoDataFile The path to the data file used by the server during proximity searches. This file contains address and location information for 2.85 million locations around the world.

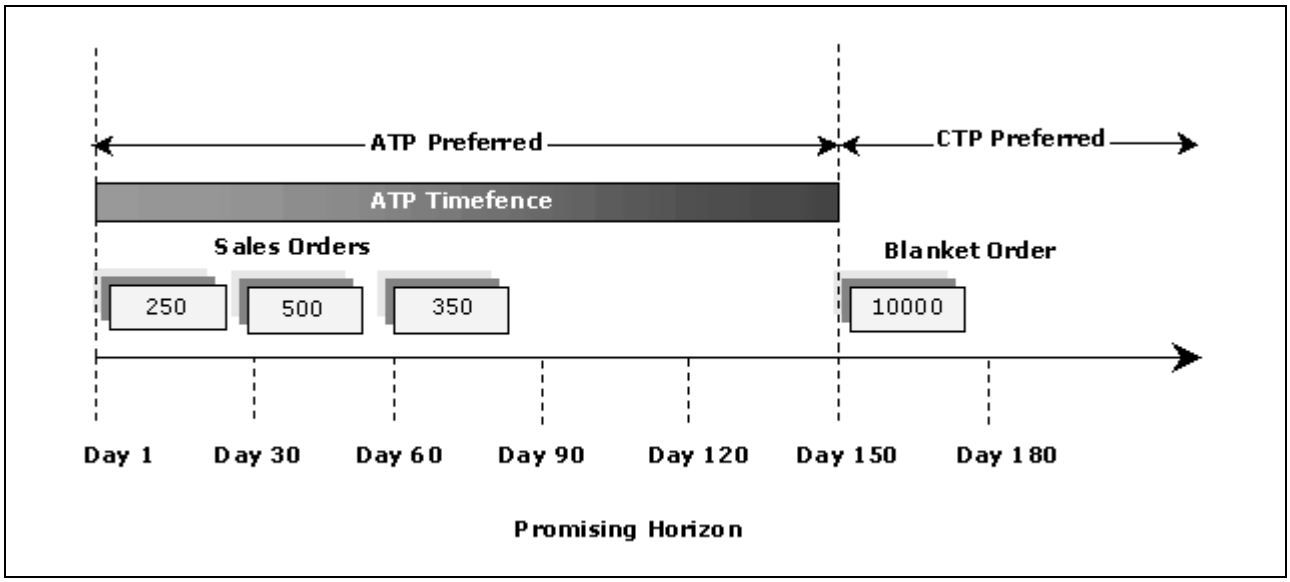
geographicCacheSize The size of the in-memory cache for geographic data. A larger cache consumes more memory but may increase the speed of the proximity search.

allocation-report	Select Y to enable the resource allocation report or select N to disable it.
atp-init	Enable or disable the generation of the initial inventory report upon server start-up. This report contains a view of the inventory levels for all items at each location in the data model for the entire promising horizon as they were calculated at the last server initialization. Inventory levels are determined based on beginning inventory, work orders, planned transfers, in-transit inventory, and previously committed customer orders. The default is Y. If you do not require this information, this flag should be set to N in a production environment.
dropdown-threshold	Specify the number of items to be displayed in drop-down boxes within the Order Promising web application. The default is 30 items. If the amount of data to be displayed is greater than the threshold set, the drop-down functionality is not available.
init-alloc-trace-depth	Specify the level of detail about the allocation of existing committed orders upon the start of the Order Promising server. This information is exported to the initialAlloc.xml file. The higher the number, the more information is stored in the log file. The default is 0.This variable is only for use by Oracle Development for debugging purposes, and will affect system performance when used.
default-query-trace-depth	Specify the level of detail about the query solving process that the system exports to the (salesOrderCode)_(serviceObjectiveCode).xml file. The higher the number, the more information is stored in the log file. The default value is 0. This variable is only for use by Oracle Development for debugging purposes, and will affect system performance when used.
post-query-atp	Specify whether the post-query inventory level is exported to the atp-post(salesOrderCode).txt file. The default is N. This variable is only for use by Oracle Development for debugging purposes, and will affect system performance when used.

Defining an ATP Timefence

An inventory policy for a product can include an ATP timefence that instructs Order Promising to use CTP capacity before using ATP capacity for orders placed outside of the ATP timefence. This timefence prevents large future orders from consuming ATP capacity that could be used for immediate orders.

For example, one of your customers has planned a sales promotion for a specific model of ceiling fan five months from now. A sales order for 10,000 items is entered that takes effect five months (or 150 days) from today's date. Meanwhile, you have firm orders for this item earlier in the promising horizon that can be allocated from available inventory of standard items. You do not want to risk blocking firm orders for a blanket order that could change in the next five months. To prevent Order Promising from blocking earlier orders, you create an ATP timefence of 150 days in the inventory policy for the item. The timefence instructs Order Promising to source the item from available inventory up until the end of the timefence. After that time, Order Promising is instructed to fulfill the order through production (or CTP capacity) until the end of the promising horizon. The ATP timefence for this item protects the orders that have been promised earlier in the promising horizon.



ATP Timefence

The ATP timefence can be set for individual items in the InventoryPolicy.xml file located in the Datastore directory. The InventoryPolicy.xml file contains information about the inventory policy for items at a location such as the item code, location code, atp timefence, inventory policy pick list, and the inventory policy purchase list. This file must be edited in XML.

To change the system-wide value for all item-location combinations, change the atpTimeFence and timeFenceUnit values in the op_Configuration.xml file located in the `\scp\8.12.1\op\connector\xsl\process` folder. The default atpTimeFence value is –1.

atpTimeFence A number that specifies the number of days from the horizon start date that Order Promising will try to promise from the existing ATP before using CTP. After that time until the end of the horizon, Order Promising will use CTP before using ATP. This attribute is only available in the Order Promising datastore; it does not originate in EnterpriseOne or SCBM. Valid values are:

0—CTP is preferred over ATP for the entire horizon.

–1—ATP is preferred over CTP for the entire horizon.

>0—The item is constrained for the specified number of days.

The default is *–1*.

Note. The InventoryPolicy.xml file gets refreshed on a daily basis from the EnterpriseOne extracts that flow through the Supply Chain Business Modeler to the Order Promising Connector, and finally, the Order Promising datastore. ATP timefence settings must be set before the Order Promising server is started to take effect.

See Also

"Appendix C: Understanding the Order Promising XML Format" for more information about how understand the Order Promising XML format.

Prioritizing the Fulfillment of Sales Orders Upon Startup

The order in which sales orders are fulfilled by the Order Promising server upon startup can be customized to ensure that critical sales orders secure available inventory and resources before other orders. Without customizing the sort order for sales orders, the Order Promising server will fulfill each sales order in chronological order upon startup. For each critical sales order, the priority attribute can be set in its header to ensure that the sales order gets priority.

The SalesOrder.xml file contains information about each sales order, including its priority. This file must be edited in XML.

priority The priority used when allocating the sales order at startup. Valid values are:

0—The sales order is not prioritized, and gets fulfilled in chronological order.

>0—The sales order is critical, and has been assigned a number to indicate its priority, with 1 being the highest priority. If sales orders are assigned the same priority, Order Promising prioritizes based on the priority code and the fulfillment dates.

The default is *0*.

Note. The SalesOrder.xml file gets refreshed on a daily basis from the EnterpriseOne extracts that flow through the Supply Chain Business Modeler to the Order Promising Connector, and finally,the Order Promising datastore. Priority settings must be adjusted daily before the Order Promising server is started to take effect.

See Also

"Appendix C: Understanding the Order Promising XML Format" for more information about how understand the Order Promising XML format.

Configuring the Datastore

This section discusses how to:

- Set datastore configuration variables.
- Define geographic database aliases.
- Verify customer addresses.

Setting Datastore Configuration Variables

Access the opdatastoreConfig.xml file.

To configure the datastore:

horizon-start-time	Specify the date and time that marks the beginning of the promising horizon.
horizon-length	Specify the length of the promising horizon in days. The default value is 365.
round-to-nearest	Enable or disable rounding of items to the nearest planning multiple. Valid values are *True* or *False.*
wo-material-purchase	Set to *True* to verify whether material demand from work orders can be satisfied by purchasing material. The demand time must be later than the purchased material's standard or premium lead time. If material cannot be purchased, the server material is consumed from existing inventory. Set to *False* to consume existing inventory to satisfy the demand from work orders. The default value is *False.*
wo-concatenate-enable	Set to *True* to enable the concatenation of fields to form manufacturing and routing codes for EnterpriseOne integration. Set to *False* if you are not integrating Order Promising with EnterpriseOne.
use-lane-constraints	Set to *True* to enable the Order Promising solver to consider the lane capacity weight and the lane transport mode calendar when fulfilling orders. Set to *False* if you do not want lane constraints to be taken into consideration.

Defining Geographic Aliases

Access the GeographicAlias.xml file.

To define a geographic alias:

1. Specify the name of the country, state/province, and city using following tags:
   - countryName
   - stateProvinceName
   - cityName
2. Specify as many aliases for each country, state/province, and city using the following flags:
   - countryAlias
   - stateProvinceAlias
   - cityAlias
3. Save the GeographicAlias.xml file.

Verifying Customer Addresses

Access a DOS or UNIX command prompt.

To verify customer addresses:

Issue the following command:

`geolookup -country-city[-admin]`

Note. The *admin* flag denotes an administrative region in a country, such as state, province, or prefecture.

Recovering the Datastore if the System Fails

This section discusses how to:

- Unlock the datastore.
- Restore the real-time EnterpriseOne messages to the datastore.

Unlocking the Datastore

To begin to restore the system, the datastore lock needs to be removed. To do so, simply delete the .dataStoreGuard file located in the Datastore directory.

Restoring the Real-time EnterpriseOne Messages to the Datastore

After the datastore has been unlocked, it is possible to restore the real-time EnterpriseOne messages from the .requestJournal file to the datastore. To do so, restart the Order Promising server. All the messages stored in the .requestJournal file are added to the datastore, and then the model is loaded into memory. Upon completion, the .requestJournal file is cleared, ready to log future real-time messages.

In some rare cases, the server might not start after a couple of attempts because the Order Promising server failure was caused by the last real-time message received from EnterpriseOne. When this happens, the Order Promising server fails again after you restore the real-time EnterpriseOne messages. Since the Order Promising server has not loaded the model into memory successfully, the .requestJournal file is still intact. Before attempting to start the Order Promising server again, it is necessary to remove the last real-time message from the .requestJournal before attempting to restart the Order Promising server.

Note. If the Order Promising server fails at the end of the day, data synchronization using the Order Promising connector can be used to update the datastore. Both the .dataStoreGuard and the .requestJournal files can be erased.

Monitoring the Server

This section describes how to:

- Monitor the server session.
- View sales orders in the server queue.

- View the slowest promises in the current session.

Page Used to Monitor the Server

Window Name	*Navigation*	*Usage*
Administration	Home Page, Administration	Monitor session promises Monitor system errors

Monitoring the Server Session

The effectiveness of promising both sales orders from EnterpriseOne as well as simulated sales orders can be monitored to determine the number of promises, average promise time, errors, and slowest promise during the current server session.

Access the Administration page.

To monitor the server session, refer to the following fields in Server Status in the Current Session:

Total Promises The total number of orders promised in the current server session.

Average Promise Time The average time it takes to promise a sales order in the current promising session.

Protocol Errors The number of data integrity errors that the system has detected as a result of data received from EnterpriseOne Supply Chain Business Modeler or through a real time integration with EnterpriseOne.

Promising Errors The number of errors that the system has detected during the order promise. Additional detail about promising errors can be found in the server log file.

Slowest Promise The slowest order promise in the current server session.

Viewing Sales Orders in the Server Queue

Access the Administration page.

To view sales orders and simulated sales orders in the promising queue, refer to the following fields in Server Queue:

Position The position of the message in the promising queue.

Message Name Messages are promising events that are received from EnterpriseOne Sales Order Entry.

Wait Time The amount of time that the promising event has been waiting in the promising queue.

Viewing the Slowest Promises in the Current Session

The slowest promises of either real-time sales orders from EnterpriseOne or simulated sales orders are displayed.

Access the Administration page.

Sales Order Number A number assigned to the sales order in EnterpriseOne Sales Order Entry.

Customer Code The customer for the sales order.

Number Of Line Items The number of individual line items that are contained in a sales order.

Created By The username of the person who entered the sales order in EnterpriseOne Sales Order Entry

Promising Time The amount of time in seconds that EnterpriseOne Order Promising required to promise the sales order.

Viewing System Configuration

This section discusses how to:

- View promise settings.
- View logging settings.
- View the server and datastore configuration.

Page Used to View System Configuration

Window Name	*Navigation*	*Usage*
Administration	Home Page, Administration	View horizon settings View logging settings View server and datastore settings

Viewing Promise Settings

Access the Administration page.

To view the promise settings, refer to the following fields in Configuration:

Horizon Start Date The date from which the current promising horizon extends. You can change this value in the datastore configuration file.

Horizon Length The length of the promising horizon in days. The length of the promising horizon combined with the start date determines the end of the promising horizon. You can change this value in the datastore configuration file.

See Also

"Setting Datastore Configuration Variables"

Viewing Logging Settings

Access the Administration page.

To view the logging settings, refer to the following fields in Configuration:

Log File Location The location of the server log file. You can determine the location of the server log file in the server configuration file.

Log File Size The size, in kilobytes, of the server log file. The size of the server log file should be monitored to ensure that you do not run out of storage space. In the event that you notice a decrease in server performance, monitor the server log file size. The log file size can be reduced by limiting the scope of errors in the log file. You can set the logging level in the server configuration file.

Log File Created On The date and time when the most recent server log file was created.

Viewing the Server and Datastore Settings

Access the Administration page.

To view the server and datastore configuration, refer to the following fields in Configuration:

License File Location The directory where the license files for EnterpriseOne Order Promising are stored.

Datastore Location The directory where the datastore resides. You can determine the directory that contains the datastore in the datastore configuration file.

Server Version The version of the EnterpriseOne Order Promising server that is currently installed on your system.

Server Location	The directory where the EnterpriseOne Order Promising server executable resides. This directory is specified during installation of the software and can not be modified.
Server Host	The hostname of the server on your network that is hosting the EnterpriseOne Order Promising server.
Server Port Number	The port number that the EnterpriseOne Order Promising server uses to communicate with other system components. The default value is 39000. You can change this value in the server configuration file.

See Also

"Configuring Server Variables"

Chapter 9

Using the Order Promising Connector

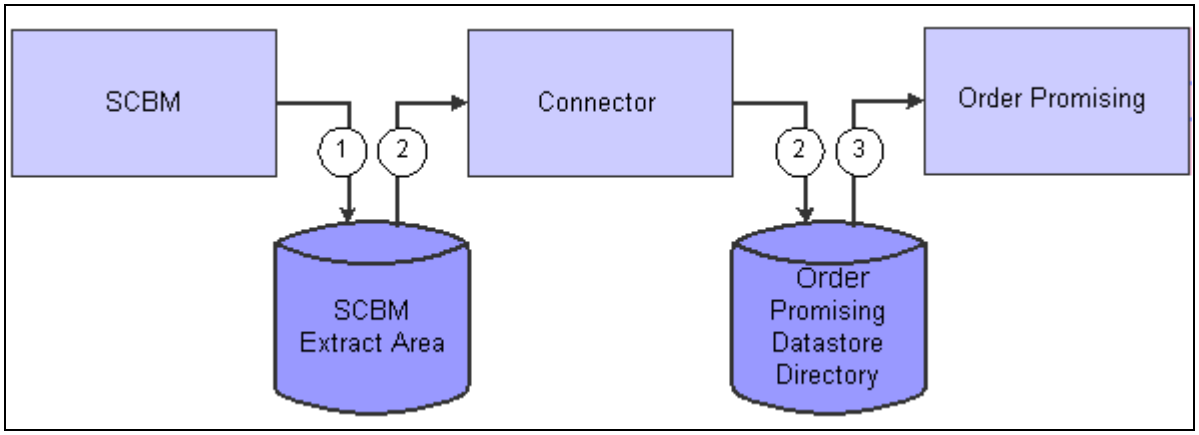
This chapter provides an overview of the process of integrating Order Promising with Supply Chain Business Modeler (SCBM) using the Order Promising Connector, and the Order Promising Connector commands. This chapter discusses how to transform the XML files exported from SCBM into the Order Promising data format.

Understanding the SCBM Integration

Using a connector that is provided with Order Promising, you can easily transfer enterprise data from SCBM to Order Promising. SCBM is a configurable supply chain data warehouse that enables you to transfer enterprise data between EnterpriseOne Supply Chain Management and Supply Chain Planning systems. After importing supply chain data into Supply Chain Business Modeler, you can export the data from the Tactical model for use in Order Promising. Using the data connector, you can convert the data and load it into Order Promising.

Note. EnterpriseOne communicates to SCBM 3.2 with the 3.0 API. The Order Promising connector uses the SCBM 3.2 API.

The following example illustrates the process of transferring data from Supply Chain Business Modeler to Order Promising:



Importing enterprise data through Supply Chain Business Modeler

To transfer data from Supply Chain Business Modeler to Order Promising, you must:

1. Export data by creating and running an export scenario. You must export the following data packages from the Tactical model:

• Base • Manufacturing • Customer • SalesOrders • Calendar • Configuration	• BeginningInventory • PurchaseOrders • TransferOrders • Supplier • WorkOrders

2. Use the refresh command to transform the enterprise data into import files that can be recognized by the datastore.
3. Import the data into Order Promising.
4. Validate the data imported against the Order Promising datastore XSD schema. This is an optional step.

Understanding the Order Promising Connector Commands

This section discusses:

- Refresh command
- Validate command

Refresh Command

Use the Refresh command to import EnterpriseOne data from SCBM into the Order Promising datastore. To assist customers to transfer large XML files from one machine or location to another, the Order Promising Connector refresh command can be customized to convert gzip compressed XML files directly from SCBM. The refresh command can also be customized to retain all the intermediate files used during the data transformation for troubleshooting purposes. The refresh command syntax is:

`op::refresh` *SCBMDataFolder OPDataFolder argument*

Parameter	***Description***
SCBMDataFolder	The location of the source SCBM data folder. If the directory name includes a space, it must be enclosed in quotation marks.
OPDataFolder	The location of the destination Order Promising datastore directory. If the directory name includes a space, it must be enclosed in quotation marks. With the standard installation, the path for this directory is: `/scp/8.12.1/op/data/opserver/datastore`.

Parameter	Description
argument	An optional argument used to customize the refresh command. Your options are: • `–debug` Use the –debug argument to retain the intermediate transformation files for debugging purposes. • `–gzip` Use the –gzip argument to allow the Order Promising Connector to accept compressed XML files.

For example, the following command converts all gzip data packages from a directory called SCBM Export to the Datastore directory:

```
op::refresh "c:/scp/8.12.1/scbm/scbm export"
"c:/scp/8.12.1/op/data/opserver/datastore" –gzip
```

Validate Command

Use the validate command to check the integrity of the transformed XML data files in the Datastore directory by comparing them against the Order Promising Datastore XSD schema. This option can help you to detect any data problems before running the Order Promising server. The syntax is:

`op::validateXml`*OPDataFolder XSDFolder*

Parameter	Description
OPDataFolder	The location of the destination Order Promising datastore directory. If the directory name includes a space, it must be enclosed in quotation marks. With the standard installation, the path for this directory is: `/scp/8.12.1/op/data/opserver/datastore`.
XSDFolder	The location of the folder containing the Order Promising XSD datastore schema. With the standard installation, the path for this directory is: `/scp/8.12.1/op/xsd`.

Prerequisite

Before you begin importing data from SCBM, ensure that you have created and run an export scenario that copies data from the Tactical model to an extract area. In addition, stop the Order Promising server when running the Order Promising connector.

Importing Enterprise Data from SCBM

Perform these steps when importing Enterprise Data from Supply Chain Business Modeler to Order Promising:

1. From a DOS command prompt or in Windows Explorer, navigate to the `/scp/8.12.1/op/bin` directory.
2. Do one of the following:
   - In Windows Explorer, double click OPConnector.exe
   - At a DOS command prompt, enter `OPConnector`

     The Order Promising connector command shell starts.
3. Enter the following commands:

   ```
   package require Aps
   ```

   The connector loads the current version of the APS package.
4. Enter the following command:

   ```
   package require Op
   ```

   The connector loads the current version of the Order Promising package.
5. Enter the following command:

   `op::refresh` *SCBMDataFolder OPDataFolder argument*

   The connector transforms the SCBM data into the Order Promising data format.
6. Optional. Enter the following command:

   `op::validateXml` *OPDataFolder XSDFolder*

   The transformed data is validated against the Order Promising XSD schema format to detect any errors.

Appendix A

Understanding Real-time Message Mapping

This appendix discusses the mappings between EnterpriseOne and Supply Chain Planning for the following real-time processes:

- Order Promising queries and replies
- Order Promising server and datastore updates

Understanding the Mappings

The Order Promising Server expects to receive a specific set of fields when processing sales order inquiries that originate from EnterpriseOne Supply Chain Management. These fields must then map to the corresponding fields in the Order Promising datastore.

See Also

"Appendix C: Understanding the Order Promising XML Format"

Mappings for SalesOrderQuery

To determine the best available date for shipment, the callOPSalesQueryBusinessService (BSFN B4205000) calls the getOrderPromising operation to get the best promised dates from the OP Web Service. Sales order information (including the header and its details) are passed to the OP Server to get the best promised dates.

Fields in Input SalesOrderQuery

The following tables describe the summary and detail fields:

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
Header			

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
szUserID	userId	string	A unique code, assigned by the external system, which identifies the user who is making the request. An entry in this field ensures that the query is routed properly to the appropriate scenario manager.
variable:maxResults	maxResults	integer	The maximum number of results that will be returned by the server. For EnterpriseOne, this field should always contain "1" or be left blank so that the default value is used.
	salesOrder Object		
szOrderNumber	salesOrderCode	string	A unique system-generated number that identifies the order.
mnCustomerId	customerCode	string	The unique code number that identifies the customer.
szCustomerName	customerName	string	The name of the customer for whom the order is being placed.
szCustomerGroup	customerGroup	string	The group to which the customer is assigned.
szCity	city	string	The city where the customer is located.
szStateProvince	stateProvince	string	The state or province where the customer is located.
szCountry	country	string	The country where the customer is located.
szBusinessObjective	serviceObjective	string	The service objective used during the sales order inquiry. Service objectives are set up in the Manage Service Objectives screen within the Order Promising Workshop.
nAllowMultiSource	allowMultiSource	boolean	A code that specifies whether the acquisition of line items from multiple sources is allowed. This field accepts these codes: true, false, 1 or 0. The default is true.
nPartialOrderShipment Allowed	allowPartialOrder Shipment	boolean	A code that specifies whether the shipment of partially filled orders is allowed. This field accepts these codes: true, false, 1 or 0. The default is true.
mnPenaltyCost Adjustment	penaltyCost Adjustment	integer	The penalty cost associated with the order. The default is 100.

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
CALC_EARLIEST_ ARRIVE_DATE = false	calcEarliestArriveDate	boolean	Whether the earliest arrival date should be calculated. The system accepts these codes: true, false, 1 or 0. The default is false.
mnTraceDepth	traceDepth	integer	Determines the volume of tracing information that the Order Promising Server writes out during the solve. The default is 0.
mnLastLineNumber	lastLineNumberUsed	double	The last line number used for the sales order. This number is used when adding or splitting lines.
mnLineNumberIncrement	lineNumberIncrement	double	The number used to increment the line numbers. This number is used in conjunction with the next line number when adding or splitting multiple lines.
detail[]	*salesOrderDetail object*		
mnLineNumber	lineItem	string	A unique number that identifies the sales order line item.
mnCacheLineNumber	cacheLineItem	string	A unique identification code that identifies the line number.
mnItemNumber	item	string	A code that identifies the item being ordered.
szPlanningUnit	planningUnit	string	The standard planning unit of the item or a variation of the item that is defined in the item master of the unit conversion tables. This field is necessary because the order might contain different units of measure than the one you use for planning. The planning unit of measure is defined in the EnterpriseOne Integration Constants.
mnPlanningQuantity	planningQuantity	double	The order quantity converted to the planning unit of measure.

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
mnPlanningMultiple	planningMultiple	double	The multiple in which items in the order are grouped. For example, a planning multiple of 12 specifies that items are included in the order only in groups of 12. The system rounds this value to the nearest multiple. For example, if you specify a planning multiple of 10 and 37 items are in stock, Order Promising allocates 30 units and no more. You can change this behavior by editing the round_to_nearest variable in the datastore configuration file.
PlanningUnitPrice	planningUnitPrice	double	The price of a single unit of the item in the planning unit of measure.
MIN_SHIPMENT_SIZE = 0.0	minShipmentSize	double	The size of the minimum shipment using the planning unit of measure. The default is 0.
jdRequestedDate	requestDate	date	The date the customer wants the order delivered.
szShippingGroup	shippingGroup	string	The number of the shipping group for the items in the order, if those items are to be delivered on the same day. If the PartialLineShipmentAllowed field is set to No, the Order Promising server automatically ships all lines together. If the PartialLineShipmentAllowed field is set to Yes, you can set up groups of items that must arrive together.
szMultiSource	multiSource	string	A code that allows or prohibits the acquisition of the line item from multiple locations. The field can also be used to define a sourcing group. Valid values are: Yes-allow multisourcing No-do not allow multisourcing Any other value-sourcing group
nAllowPartialLineShip	allowPartialLineShip	boolean	A code that allows or prohibits the shipment of individual line items as they become available. The valid values are: true, false, 1 or 0. The default is true.
nAllowBackorders	allowBackOrders	boolean	A code that allows or prohibits items that are not currently in stock to be promised when they become available. Valid values are: true, false, 1 or 0. The default is true.

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
nAllowSubstitutions	allowSubstitutions	boolean	A code that specifies whether to allows or prohibit the substitution of items when the original choice is unavailable. Valid values are true, false, 1 or 0. The default is true.
ASAP_ORDER = false	asapOrder	boolean	Indicates whether the customer wants the order fulfilled as soon as possible. Valid values are true, false, 1 or 0. The default is false.
szCity	city	string	Name of the city (optional). If provided, it overrides the value on the header for this line item.
szState	stateProvince	string	The name of the state or province (optional). If provided, it overrides the value on the header for this line item.
szCountry	country	string	The name of the country (optional). If provided, it overrides the value on the header for this line item.
mnConfigurationId Number mnComponentIdNumber mnParentIdNumber	plrId	string	A unique number representing a concatenation of the mnConfigurationIdNumber, mnComponent IdNumber, and mnParentIdNumber.
detail RLR[]			
mnConfigurationId Number mnComponentIdNumber mnParentIdNumber	plrId	string	A unique number representing a concatenation of the mnConfigurationIdNumber, mnComponent IdNumber, and mnParentIdNumber.
mnWorkOrderNumber	workOrderCode	string	A code that identifies the work order. The code must be unique for each location.
szBranchPlant	location	string	A code that identifies the location where the current work order is defined and, implicitly, the location of the manufacturing process.
cWOChanges Allowed	workOrderChanges Allowed	boolean	A code that indicates whether changes to the work order are allowed. Valid values are: 1 or True. Allow changes to the work order. 0 or False. Do not allow changes to the work order.

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
detail RLR Routing[]	*routingStep Object*		
szBranchPlant mnItemNumber szTypeOfRouting mnBatchQuantity mnOperationSequence szTypeOperationCode szLineCellIdentifier jdEffectiveFromDate szWorkCenter	operationCode	String	A unique code that identifies a manufacturing operation. This field is a concatenation of the szBranchPlant, mnItemNumber, szTypeOfRouting, mnBatchQuantity, mnOperationSequence, szTypeOperationCode, szLineCellIdentifier, and szWorkCenter fields.
mnOperationSequence	operationSequence	integer	A unique number within a manufacturing routing that identifies the order of operations.
mnSuccessiveOperation	successiveOperationSequence	integer	A number that specifies an operation instance that follows in sequence after the current operation instance. If the current operation is the last operation in the sequence and has no successive operation, then the value is 0.
mnPrecedenceOffset	precedenceOffset	double	The time offset between the start and end of the current operation and the start and end of the next operation. The meaning depends on the PrecedenceType field value. Valid values are: • Sequence • StartToStart • StartToEnd • EndToStart • EndToEnd
szPrecedenceType	precedenceType	string	The type of the precedence relationship between the current and the next operation.
mnQueueHours	queueTime	double	The separation time that is used as a waiting time due to specific business reasons, before the system runs the current manufacturing step specified by RoutingId. QueueHours is defined before SetupHours. This number is optional.

Business Service Value Object Field Name	***OP WSDL Fields***	***OP Data Type***	***Description***
mnSetupHours	setupTime	double	The separation time used to model any setup activity that might be required to run the manufacturing step specified in the RoutingId field. The value is optional.
mnMoveHours	moveTime	double	The separation time used to model the inventory moving activity that might be required after the manufacturing step specified in the RoutingId field is completed. The inventory move occurs even if there is no SuccessiveOperation that needs to be executed after the current manufacturing step. This value is optional.
detail RLR Resource List[]	*part object*		
mnShortItemNumber	partCode	string	A code that identifies the product.
szResourceType	partType	string	A code that identifies the type of part. Valid values are: Item PrimaryOutput DurationResource Crew Machine Tool CoProduct
szResourceId	partId	string	A code that identifies the part on the work order. This field is a concatenation of the ConfigurationIdNumber and the ComponentIdNumber fields.
mnQuantityPerPlanned	quantity	double	The quantity of product requested for this order.
PlanningUnitOf Measure	quantityUnit	string	The unit of measure used for planning.
mnConfigurationID Number mnComponentIDNumber mnParentIDNumber	plrID	string	Concatenation of mnConfigurationIDNumber, mnComponentIDNumber, and mnParentIDNumber.

Fields in Output SalesOrderQuery

This table describes the fields in the output SalesOrderQuery business service:

Order Promising Fields	***EnterpriseOne Fields***	***EnterpriseOne Data Type***	***Description***
salesOrderQueryResult Object			
salesOrderCode	szOrderNumber	String	The reference number for the order that is assigned by EnterpriseOne.
serviceObjective	szBusinessObjective	String	The service objective used during the sales order inquiry. Service objectives are set up in the Manage Service Objectives screen within the Order Promising Workshop.
result Object			
totalCost	mnTotalCost	MathNumeric	The total cost for all of the items ordered.
totalDeliveryCost	mnTotalDeliveryCost	MathNumeric	The total cost of delivery for the order.
totalPrice	mnTotalPrice	MathNumeric	The total price of the order.
totalProfit	mnTotalProfit	MathNumeric	The amount of profit from the sale of the items.
totalMargin	mnTotalMargin	MathNumeric	The profit margin associated with shipping this order.
totalValue	mnTotal Value	MathNumeric	The total value of the order.
latestLineDate	jdLatestLineDate	Date	The latest date on which the manufacturing of an item can begin.
numberOfBackorders	mnNumberOfBackorders	MathNumeric	The number of items that are unavailable on the shipping date. These items will be shipped when they become available.
numberOfSubstitutions	mnNumberOfSubstitutions	MathNumeric	The number of product substitutions made if line items are unavailable on the shipping date.
orderFillRate	mnOrderFillRate	MathNumeric	The percentage of the order that was allocated.
numberOf AtpItems			The number of items that could be fulfilled by ATP.

Order Promising Fields	*EnterpriseOne Fields*	*EnterpriseOne Data Type*	*Description*
numberof CtpItems			The number of items that could be fulfilled by CTP.
lastLineNumberUsed	mnLastLineNumber	MathNumeric	The last line number used for the sales order. This number is used when adding or splitting lines.
lineNumberIncrement	mnLineNumberIncrement	MathNumeric	The number used to increment the line numbers. This number is used with the next line number when adding or splitting multiple lines.
detail Object			
lineItem	mnLine Number	MathNumeric	The line number that Order Promising assigns to the order. In Order Promising, line items can be split. When you do so, Order Promising keeps the original line item, but decrements it as a decimal - for example, line item 1.000 will be split into the following three lines: 1.000, 1.001, 1.002.
originalLineItem	mnOriginal Line Number	MathNumeric	The line number that was assigned to the order by EnterpriseOne.
cacheLineItem	mnCacheLineNumber	MathNumeric	A unique code identifying the sales order line number.
requestedItem	mnRequestedItem	MathNumeric	A code that identifies the item that the customer wants to order.
availableItem	mnAvailableItem	MathNumeric	A code that identifies the available item or its substitute (if product substitution is allowed).
availableAmount	mnAvailableAmount	MathNumeric	The quantity of the line item that is available on the shipping date.
quantityUnit			The unit of measure for the line item quantity.
availableDate	jdAvailableDate	Date	The date that the item arrives at the customer location.
requestedDate	jdRequested Date	Date	The date the customer wants the item to be delivered.

Order Promising Fields	*EnterpriseOne Fields*	*EnterpriseOne Data Type*	*Description*
asapOrder			Indicates whether the customer wants the order fulfilled as soon as possible. Valid values are true, false, 1 or 0. The default is false.
earliestAriveDate			The earliest arrival date that the customer will accept the order.
shipDate	jdShipDate	Date	The date that the item ships from its final distribution point.
pickDate	jdPickDate	Date	The date when the item is prepared for shipment.
shipLocation	szShipLocation	String	The location from which the line item is shipped.
cost	mnCost	MathNumeric	The total cost of fulfilling this particular line item.
deliveryCost	mnDeliveryCost	MathNumeric	The cost to deliver this line item.
price	mnPrice	MathNumeric	The total extended price of the line.
profit	mnProfit	MathNumeric	The projected profit generated from the shipment of this particular line item.
margin	mnMargin	MathNumeric	The profit margin associated with shipping this line item.
value	mnValue	MathNumeric	The total value of the shipment.
lineFillRate			Not used by EnterpriseOne.
parentFillRate			Not used by EnterpriseOne.
substitutionRatio	mnSubstitutionRatio	MathNumeric	The ratio of the substituted item used for each unit of the original product or material.
allowPartialLineShip	nAllowPartialOrderShip	Integer	A code that specifies whether line items can be shipped as they become available.
allowBackOrders	nAllow Backorders	Integer	A code that specifies whether items that are not in stock can be promised as they become available.
allowSubstitution	nAllowSubstitution	Integer	A code that specifies whether items can be substituted when sufficient quantities of the preferred choice are not available.

Order Promising Fields	*EnterpriseOne Fields*	*EnterpriseOne Data Type*	*Description*
orphanOldworkOrders	cCancelOriginalWorkOrders	String	A code that specifies if the work order should be cancelled or not. Valid values are: 1. Cancel the work order. 0. Do not cancel the work order.
errorCode	szErrorCode	String	OP error code.
	szSuspectedCause	String	Maps OP error code to OWError from F34A50.
plrId and plrResult			
plrID	mnConfigurationIDNumber mnComponentIDNumber mnParentComponentIDNumber	MathNumeric	The unique configuration ID number representing the concatenation of mnConfigurationIDNumber, mnComponentIDNumber and mnParentComponentIDNumber.
startDate	jdStartDate	Date	The date when the operation step must be started.
endDate	jdRequestDate	Date	The date when the operation step must be completed.
itemCode	mnShortItemNumber	MathNumeric	A user-defined field that contains an alphanumeric code for the item.
locationCode	szBranchPlant	String	A code that identifies the business unit, cost center, branch, or plant.
workOrderCode	mnWorkOrderNumber	MathNumeric	A code that identifies the work order.
originalLine Number	mnOriginalLineNumber	MathNumeric	A number that links a promised sales order line with an original sales order line. The responding system preserves this value from the original request.
quantity	mnQuantity	MathNumeric	The quantity of product that Order Promising can fulfill for the work order.
promising Fault Object			
code	szErrorCode szErrorDescription	String	An error code returned by Order Promising when an error occurs.
description		String	mapping of OWError to szErrorDescription

Mappings to Update the Datastore

When the customer service representative commits the sales order, the system sends the sales order details to the Order Promising server. Work order and the work order bill of material details are also sent with any sales orders that contain configured items. Finally, any changes to sales orders, purchase orders, transfer orders, and manual inventory are also sent to the Order Promising server in real time.

Note. Updates to work orders and work order parts lists and routings are also sent to Order Promising as they occur, however, they are not included in the Order Promising model until the Order Promising server is restarted.

The web service operations that update the Order Promising model are:

- AdjustInventoryNotify (business service processAdjustInventory)
- ProcurementNotify (business service processProcurement)
- SalesOrderNotify (business service processSalesOrder)
- WorkOrderNotify (business services processWorkOrder and processBOMR)

Fields in AdjustInventoryNotify

This table describes the fields in the AdjustInventoryNotify, supported by the processAdjustInventory business service:

Business Service Value Object Field Name	*Order Promising WSDL Field*	*OP Data Type*	*Description*
cActionCode	action	string	The action code specifies the type of change that was made to the item balance table. Valid values are: 1. add 2. change 3. delete
szLotNumber	lotNumber	string	The unique code identifying the lot at the location. This code is a concatenation of the EnterpriseOne fields Location and Lot Number.
szBranchPlant	locationCode	string	The unique code identifying the inventory location.
mnShortItemNumber	itemCode	string	The unique code identifying the item stored at the location.
mnPlanningQuantity	quantity	double	The amount of the item contained in this lot. This field is mandatory.

Business Service Value Object Field Name	Order Promising WSDL Field	OP Data Type	Description
szPlanningLotStatus	lotStatus	op:LotStatus	The status of the lot. Valid values are: Available-The lot is available. Scrap-The lot has been scrapped and cannot be used. This might be due to damage, breakage, spoilage, and so on. OnHold-The lot is on hold for some reason, such as quality control, quarantine, or curing. Pegged-This field is reserved for future use. Expired-The lot has expired and cannot be used. The lot status values are mapped from the 34A/LS UDC.
	statusDate	date	This field is not used in EnterpriseOne.
	manufacturingDate	date	This field is not used in EnterpriseOne.
	holdPeriod	integer	This field is not used in EnterpriseOne.

Fields in ProcurementNotify

This table describes the fields in ProcurementNotify, supported by the processProcurement business service:

Business Service Value Object Field Name	Order Promising Datastore Field	OP Data Type	Description
	shipment Object		
cOrderAction	action	string	The action code specifies the type of change made to the purchase order or transfer order field. Valid values are: 0. do nothing 1. add 2. change 3. delete
szBranchPlant	destination Location	string	The business unit, cost center, branch, or plant the order is being shipped to.
jdOrderDate	orderDate	date	The date the order was placed.
szShippingBranchPlant	originLocation	string	The shipping branch or plant from which the shipment originates.

Business Service Value Object Field Name	***Order Promising Datastore Field***	***OP Data Type***	***Description***
mnOrderNumber szOrderType szOrderCompany szOrderSuffix	transferOrderNumber	string	The unique code that identifies the shipment. The code is a concatenation of the following fields: mnOrderNumber, szOrderCompany, szOrderType, and szOrderSuffix.
detail.cTransfer DirectShipFlag	type	string	Identifies the order as either a transfer or purchase order. Valid values are: • TransferOrder • PurchaseOrder
detail[]	*procurementItem Object*		
cOrderAction	action	string	The action code indicates whether an item has been added, changed, or deleted from a transfer order. Valid values are: 1. add 2. change 3. delete
jdActualShipDate	actualShipDate	date	The actual date the item was shipped from the warehouse.
mnPlanningQuantity mnQuantityReceived Planning	currentOrderQuantity	double	The order quantity. This field is a sum of mnPlanningQuantity and mnQuantityReceivedPlanning.
szItemNumber	itemCode	string	The short item number of the purchase order line.
jdPromisedDeliveryDate	plannedArrivalDate	date	The date that an item will be delivered to the customer.
jdPromisedShipDate	plannedShipDate	date	The date that the item can be shipped from the warehouse.
szPlanningUOM	quantityUnit	string	The unit of measure used for planning.
mnOrderLineNumber	transferOrderItemNumber	string	The detail line number.
szModeOfTransport	transportMode	string	The code that describes the transportation means (for example, by rail).
"Planned Transfer"	type	string	This field is hard coded with the value "PlannedTransfer".

Fields in SalesOrderNotify

This table describes the fields in SalesOrderNotify, supported by the processSalesOrder business service:

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
	salesOrder Object		
cOrderAction	action	op:HeaderNotificationAction	The Action Type field specifies whether a sales order has been added, changed, or deleted from EnterpriseOne. Valid values are: 0. do_nothing 1. add 2. change 3. delete
mnOrderNumber sz OrderType szOrderCompany	salesOrderCode	string	The sales order number. This field is a concatenation of mnOrderNumber, szOrderType, and szOrderCompany.
mnShipToAddress Number	customerCode	string	The address book number of the person to whom the item is to be shipped.
szShipTo MailingName	customerName	string	The SoldTo address book number.
	customerGroup	string	Not mapped in EnterpriseOne.
szShipToAddressLine1	address1	string	The first line of the address record.
szShipToAddressLine2	address2	string	The second line of the address record.
szShipToAddressLine3	address3	string	The third line of the address record.
szShipToCity	city	string	The name of the city to which the order is to be shipped.
szShipToCounty	county	string	The name of the county to which the order is to be shipped.
szShipToState	stateProvince	string	The name of the state or province to which the order is to be shipped.
szShipToCountry	country	string	The name of the country to which the order is to be shipped. This field is not mapped in EnterpriseOne.

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
szShipToZipCode	postalCode	string	The zip or postal code to which the order is to be shipped..
OPBusiness Objective	serviceObjective	string	The service objective associated with the sales order.
	allowMultiSource	boolean	This field indicates whether multiple sources are allowed for this sales order. The default for this field is "false".
	penaltyCostAdjustment	integer	The penalty cost if the order is not fulfilled by the customer request date. The default for this field is 0.
	allowPartialOrder Shipment	boolean	This field indicates whether partial shipment is allowed for this sales order. The default for this field is "true".
	priority	integer	This field indicates the priority of the order
detail[]	*salesOrderDetail Object*		
cOrderAction	action	op:DetailNoti-ficationAction	The Action Type specifies whether the sales order item has been added, changed, or deleted. Valid values are: 1. Add 2. Change. 3. Delete
mnLineNumber	lineItem	string	The sales order line number.
mnShortItemNumber	itemCode	string	The short item number of the sales order line.
mnPlanningQuantity	quantity	double	The quantity used for planning.
szPlanningUnitOf Measure	quantityUnit	string	The unit of measure used for planning.
jdRequestedDate	requestedDate	date	The date the item has been requested.
jdPromisedShipDate	shipDate	date	The date that the item can be shipped from the warehouse.
"Approved"	status	op:Status	This field is hardcoded with the value "Approved".
jdScheduledPickDate	pickDate	date	The day that the item can be picked up from the warehouse.

Business Service Value Object Field Name	Order Promising WSDL Field	OP Data Type	Description
szDetailBranchPlant	shipFromBranchCode	string	The business unit, cost center, branch, or plant from which the item is shipped.
jdPromisedDeliveryDate	arriveDate	date	The date that an item will be delivered to the order company.
	city	string	The name of the city to which the order is to be shipped.
	county	string	The name of the county to which the order is to be shipped.
	stateProvince	string	The name of the state or province to which the order is to be shipped.
	country	string	The name of the country to which the order is to be shipped. This field is not mapped in EnterpriseOne.
	postalCode	string	The zip or postal code to which the order is to be shipped..
szShipComplete	allowPartialLineShip	boolean	This field indicates whether partial shipment is allowed for this sales order item.
cBackOrdersAllowed	allowBackOrders	boolean	This field indicates whether backorders are allowed for this sales order item.
	allowSubstitutions	boolean	This field indicates whether substitutions are allowed for this sales order item.
	allowMultiSource	boolean	This field indicates whether multiple sources are allowed for this sales order item.

Fields in WorkOrderNotify

This table describes the fields in WorkOrderNotify, supported by the processWorkOrder business service:

Business Service Value Object Field Name	Order Promising WSDL Field	OP Data Type	Description
cEventNotificationAction Code	action	op:WorkOrder HeaderNotifica-tionAction	The code that represents the work order.

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
szOrderNumber szOrderType	workOrderCode	string	A code that represents the work order. The code is a concatenation of szOrderNumber and szOrderType.
szBranchPlant	locationCode	string	The business unit, cost center, branch, or plant where the work order is fulfilled.
	description	string	The description for the work order.
szPlanningOrderType	type	op:WorkOrder Type	A code that identifies the planning system order type. Valid values are Production, Maintenance, or Configured.
mnShortItemNumber	itemCode	string	The short item number of the work order item. This is a key field and is required for production orders.
	manufacturingCode	string	A code that identifies the manufacturing process assigned to this work order. The manufacturing process is either a routing (sequence of operations) or a single operation. This field is not used by EnterpriseOne.
mnPlanningQuantity	quantity	double	The primary output quantity when the manufacturing process is complete. This field is required for production or configured orders.
szPlanningUOM	quantityUnit	string	The unit of measure used for the quantity field. This field is required for production or configured orders.
cWOStatusFlag	status	op:WorkOrder Status	The code that describes the status of a work order. Valid values are: 1. Open 2. Open 3. Active 4. Open 5. Closed
	creationDate	date	The date that an order was entered into the system.
jdRequestDate	requestedDate	date	The date that an item is to arrive or that an action is to be complete.
jdStartDate	startDate	date	The start date for the work order.

Business Service Value Object Field Name	*Order Promising WSDL Field*	*OP Data Type*	*Description*
	completionDate	date	The completion date for the work order.
cWOChangeAllowed	changesAllowed	boolean	A code that indicates whether a work order can be changed. Valid values are: Y. Changes are communicated to Order Promising. N. Changes are not communicated to Order Promising.
szParentOrderNumber	configuredParentWork Order	string	The code for the configured parent work order. This field is used to identify the parent work order for configured items that have configured sub assemblies with a separate work order.
	configuredParentLocation	string	The code for the location where the configured parent work order is set to run. Configured sub assemblies might be produced at different locations than the parent.
mnRelatedOrderNumber szRelatedOrderType szRelatedOrderCompany	salesOrderCode	string	The configured parent sales order number. The order number is used to update the WorkOrder object in the Order Promising datastore. This field is a concatenation of mnRelatedLineNumber , szRelatedOrderType, and szRelatedOrderCompany.
mnRelatedLineNumber	salesOrderLineItem	string	The configured parent sales order line item number derived from the order number. This data is used to update the WorkOrder object in the Order Promising datastore.

Fields in WorkOrderNotify (Parts and Routings)

This table describes the fields in WorkOrderNotify, supported by the processBOMR business service:

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
cEventNotificationActionCode	action	op:WorkOrderHeaderNotificationAction	The Action Type field specifies whether a sales order has been added, changed, or deleted from EnterpriseOne. Valid values are: 0. do_nothing 1. add 2. change 3. delete
mnOrderNumber szOrderType	workOrderCode	string	A code that represents the work order. This field is a concatenation of mnOrderNumber and szOrderType.
szBranchPlant	locationCode	string	The business unit, cost center, branch, or plant where the work order is fulfilled.
szWODescription	description	string	A description of the work order.
szPlanningOrderType	type	op:WorkOrderType	A code that identifies the planning system order type. Valid values are Production, Maintenance, or Configured.
mnShortItemNumber	itemCode	string	The short item number of the work order item. This is a key field and is required for production orders.
	manufacturingCode	string	A code that identifies the manufacturing process assigned to this work order. The manufacturing process is either a routing (sequence of operations) or a single operation. This field is not used by EnterpriseOne.
mnPlanningQuantity	quantity	double	The primary output quantity when the manufacturing process is complete. This field is required for production or configured orders.
szPlanningUOM	quantityUnit	string	The unit of measure used for the quantity field. This field is required for production or configured orders.

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
cWOStatusFlag	status	op:WorkOrder Status	The code that describes the status of a work order. Valid values are: 1. Open 2. Open 3. Active 4. Open 5. Closed
jdTransactionDate	creationDate	date	The date that an order was entered into the system.
jdRequestDate	requestedDate	date	The date that an item is to arrive or that an action is to be complete.
jdStartDate	startDate	date	The start date for the work order.
cWOChangeAllowed	changesAllowed	boolean	A code that indicates whether a work order can be changed. Valid values are: Y. Changes are communicated to Order Promising. N. Changes are not communicated to Order Promising.
szParentOrderNumber	configuredParentWork Order	string	The code for the configured parent work order. This field is used to identify the parent work order for configured items that have configured sub assemblies with a separate work order.
mnRelatedOrderNumber szRelatedOrderType szRelatedOrderCompany	salesOrderCode	string	The configured parent sales order number. The order number is used to update the WorkOrderobject in the Order Promising datastore. This field is a concatenation of mnRelatedOrderNumber, szRelatedOrderType, and szRelatedOrderCompany.
mnRelatedLineNumber	salesOrderLineItem	string	The configured parent sales order line item number derived from the order number. This data is used to update the WorkOrder object in Order Promising.
szParentWOBranchPlant	configuredParentLocation	string	The code for the location where the configured parent work order is set to run. Configured subassemblies might be produced at different locations than the parent.
routing[]	*workOrderRouting*		

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
cActionType	action	op:WorkOrder HeaderNotifica-tionAction	The Action Type specifies whether the work order routing has been added, changed, deleted, or replaced. Valid values are: 1. Add 2. Change. 3. Delete 4. Replace 5. Do Nothing
mnOperationSequence Number	operationSequence	integer	A unique number within a manufacturing routing that identifies the order of operations.
OPERATION_CODE= Blank	operationCode	string	The operationCode is blank for both configured and non-configured work orders.
mnSuccessiveOperation	successiveOperation Sequence	integer	The next operation in the routing sequence.
mnQueueHours	queueTime	double	The total hours that an order is expected to be in queue at work centers and moving between work centers.
	queueTimeUnit	string	The unit of measure for the setup, move, and queue times. The default unit is hours.
mnSetupHours	setupTime	double	The standard setup hours that are incurred in the normal completion of this routing step.
	setupTimeUnit	string	The unit of measure for the setup, move, and queue times. The default unit is hours.
mnMoveHours	moveTime	double	The planned hours required to move the order from the current operation to the next.
	moveTimeUnit	string	The unit of measure for the setup, move, and queue times. The default unit is hours.

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
szPrecedenceType	precedenceType	op:Precedence Type	The type of the precedence relationship between the current operation and the next operation. Valid values are: • Sequence • StartToStart • StartToEnd • EndToStart • EndToEnd The default value is Sequence.
mnPrecedenceOffset	precedenceOffset	double	The time offset between the start and end of the current operation and the start and end of the next operation. The meaning depends on the precedence_type field value. Values are optional; the default value is 0.0.
	status	string	The status of the work order. Valid values are: 1. Open 2. Open 3. Active 4. Open 5. Closed This field is optional.
jdRequestDate	requestedDate	date	The date that the routing step is to be complete.
jdPlannedStartDate	plannedStartDate	date	The date that the work order is planned to start.
jdPlannedFinishDate	plannedFinishDate	date	The date that the work order is planned to be completed.
	actualStartDate	date	The actual date that the work order was started. This field is optional.
	actualFinishDate	date	The actual date that the work order was finished. This field is optional.
part[]	*workOrderPart*		

Business Service Value Object Field Name	***Order Promising WSDL Field***	***OP Data Type***	***Description***
cActionType	action	op:DetailNoti-ficationAction	The type of net change action. Valid values are: 1. Add 2. Change 3. Delete If the header action is "replace", the detail action is "add".
szResourceId	bomrId	string	A code that uniquely identifies a resource. For crew, machine, and tool, and primary output resources, this field is a concatenation of the: • Operation Sequence Number • Work Center • Operation Type • Resource Type Identifier - (Resource Line Number for crew, machine, and tool resources. 'P' for Primary Output) For configured component resources this field is a concatenation of the: • Configuration Id • Configuration Component Id For non-configured component resources this field is the parts list unique key.
Either szShortItemNumber, or szResourceCode	partCode	string	If the ResourceType is a primary output or item, then the Resource Id is the Short Item Number. If the Resource Type is a duration resource, crew, machine or tool, then the partCode is the Resource Code.
	partDescription	string	The description of the resource list item.
szResource Type	partType	op:PartType	A value that defines the role of the part.
mnQuantityPlanned	totalQuantity	double	The total amount of the item that is produced or consumed upon completion of the work order.

Business Service Value Object Field Name	*Order Promising WSDL Field*	*OP Data Type*	*Description*
mnQuantityPlanned	remainingQuantity	double	The remaining quantity of the item that needs to be produced or consumed to complete the work order. This quantity is determined when the current operation has begun, but has not been completed.
mnQuantityPer	quantityPer	double	The quantity of bill of material and resource component that is required to make one unit of work order output. The scrap and yield factors are inferred. This field is optional.
szPlanningUnitOf Measure	quantityUnit	string	The unit of measure used to define the Quantity Planned. The default is hours.
DEFAULT_ CONSUMPTION_TYPE = "Variable"	consumptionType	op:Consump-tionType	The consumption type for the bill of material and resource component. Valid values are: 0. The consumption is fixed. Order Promising consumes a fixed amount of resources and materials regardless of the number of units of output. 1. The consumption is variable. Order Promising consumes a variable amount of resources and materials based on the quantity required to make each unit of output. For example, to produce 10 units of output product, Order Promising needs to consume a quantity 10 times larger than the quantityPer value. The default value is 1.
	yield	double	Optional. Defaults to 1.
	scrap	double	Optional. Defaults to 0.
mnSubassemblyWork OrderNumber	configuredSubassembly WorkOrder	string	The work order code for the configured subassembly. This field is optional.
szComponentBranch	configuredSubassembly Location	string	The plant location where the configured subassembly is manufactured. This field is optional.

Appendix B

Understanding the Supply Chain Planning XML Format

This appendix discusses:

- Supply Chain Planning XML format
- Location of the XML schema and sample data

Understanding the Supply Chain Planning XML Format

The XML data exchanged between EnterpriseOne and Supply Chain Planning Supply Chain Business Modeler is in Extensible Markup Language (XML) format. Supply Chain Business Modeler uses the XML version 1.0 standard that is officially recommended by the World Wide Web Consortium as of 1998. Unlike flat file data that uses tabs or other characters as content delimiters, data in XML format uses tags to define the data.

EnterpriseOne 8.12 SP1 and Supply Chain Business Modeler 8.12.1 exchange data using an XML format called Supply Chain Planning XML 3.0 format, which has been developed for integrating EnterpriseOne supply chain products. In Supply Chain Planning XML format, data is divided into separate XML documents, or packages. Each package includes related data that must be stored and transferred together to ensure that the data is consistent and reliable. For example, the Manufacturing package includes related information about operations, routings, and resources.

For more information about Supply Chain Planning XML format, you can view XML schema definitions. XML schema definitions describe valid data package formats, including the elements that can appear, the order of the elements, and the valid data values in each package.

EnterpriseOne Supply Chain Business Modeler is shipped with XML schema definitions that describe data packages for full import scenarios and for incremental import scenarios. Because data in incremental import scenarios is merged with existing model data, data packages for incremental scenarios do not require all data values that are required in full import scenarios.

This table indicates the locations where you can find XML schema definitions for Supply Chain Planning XML format:

XSD	***Location***
Supply Chain Planning XML 3.0- Full import scenarios	In Windows:*path*\SCP\8.12.1\SCBM\docs\xsd\3.0\full*model_type* In UNIX:*path*/SCP/8.12.1/SCBM/docs/xsd/3.0/full/*model_type*

XSD	*Location*
Supply Chain Planning XML 3.0- Incremental import scenarios	In Windows: *path* \SCP\8.12.1\SCBM\docs\xsd\3.0\incremental*model_type* In UNIX:*path* /SCP/8.12.1/SCBM/docs/xsd/3.0/incremental/*model_type*

Note. *path*is the drive where SCBM is installed and*model_type*is the type of SCBM model that you are importing data into or exporting data from.

You can also view sample data packages in Supply Chain Planning XML 3.0 format for full import scenarios. Sample data packages are saved in the*path*`/SCP/8.12.1/SCBM/sample_data/`*model_type* directory in Windows and the *path*`/SCP/8.12.1/SCBM/sample_data/`*model_type* directory in UNIX, where *path* is the directory where SCBM is installed and *model_type* is the type of SCBM model that you are importing data into or exporting data from.

XML Schema Definition

This sample includes annotated excerpts from a Base package XML schema definition:

```
<!-- Specify that the document uses XML version 1.0 and the     -->
<!-- UTF-8 character set. (SCBM can import files that use any    -->
<!-- character set supported by the Xerces XML parser, including  -->
<!-- UTF-8, ISO-8859-1, ASCII, EBCDIC, UTF-16, and Win-1252.)    -->
<!-- Specify that elements and data types come from the       -->
<!-- http://www.w3.org/2001/XMLSchema namespace and that elements -->
<!-- from this namespace begin with xs:               -->

<?xml version=1.0 encoding=iso-8859-1?>

 <xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>

<!-- Specify that the root element of the XML document is a complex   -->
<!-- element called scbm-extract. In this example, this element can   -->
<!-- include itemList, standardUomList, and itemUomList elements.    -->
<!-- Because maxOccurs defaults to 1 and minOccurs=0 for these     -->
<!--⇒
 elements, itemList, standardUomList, and itemUomList can      -->
<!-- appear one or no times in the XML document. The sequence element  -->
<!-- indicates that if the itemList, standardUomList, and        -->
<!-- itemUomList elements appear, they must appear in the order     -->
<!-- specified. The scbm-extract element must have a version      -->
<!-- attribute with a value of scp 3.0.                 -->

<xs:element name=scbm-extract>
  <xs:complexType>
    <xs:sequence>
        <xs:element name=provenance type=provenanceType
        minOccurs=0 maxOccurs=1/>
        <xs:element name=itemList type=itemListType
        minOccurs=0/>
        <xs:element name=standardUomList type=standardUomListType
        minOccurs=0/>
        <xs:element name=itemUomList type=itemUomListType
        minOccurs=0/>
    </xs:sequence>
    <xs:attribute name=version type=xs:string fixed=scp 3.0 use=required/>
  </xs:complexType>

</xs:element>

<!-- Specify that elements in the XML document with the provenanceType -->
<!-- type can include source, comment and timestamp elements. The   -->
<!-- source and comment elements have the scbmString type. The -->
<!-- timestamp element has the scbmDT type.

<xs:complexType name=provenanceType>
  <xs:all>
    <xs:element name=source type=scbmString minOccurs=0 maxOccurs=1 nillable=true/>
    <xs:element name=comment type=scbmString minOccurs=0 maxOccurs=1
       nillable=true/>
    <xs:element name=timestamp type=scbmDT minOccurs=0 maxOccurs=1 nillable=true/>
  </xs:all>
</xs:complexType>

<!-- Specify that elements in the XML document with the itemListType  -->
<!-- type can include any number of item elements with the     -->
<!-- itemObject type.                        -->

   <xs:complexType name=itemListType>
     <xs:sequence>
        <xs:element name=item type=itemObject minOccurs=0 maxOccurs=unbounded />
     </xs:sequence>
   </xs:complexType>
```

```
<!-- Specify that elements in the XML document with the      -->
<!-- standardUomListType type can include any number of      -->
<!-- standardUom elements with the standardUomObject type.     -->

<xs:complexType name=standardUomListType>
  <xs:sequence>
    <xs:element name=standardUom type=standardUomObject minOccurs=0
       maxOccurs=unbounded />
  </xs:sequence>
</xs:complexType>

<!-- Specify that elements in the XML document with the itemUomListType -->
<!-- type can include any number of itemUom elements with the      -->
<!-- itemUomObject type.                        -->

<xs:complexType name=itemUomListType>
  <xs:sequence>
    <xs:element name=itemUom type=itemUomObject minOccurs=0 maxOccurs=unbounded />
  </xs:sequence>
</xs:complexType>

<!-- Specify that elements with the itemObject type can include   -->
<!-- itemCode, itemName, alternateItemId, description, planningUom, -->
<!-- shippingUom, weight, weightUom, volume, volumeUom and     -->
<!-- storageRequirement elements. The weight and volume elements  -->
<!-- have the scbmDouble type. The remaining elements have the   -->
<!-- scbmString type. xs:all specifies that these elements can   -->
<!-- appear in any order. minOccurs=1 specifies that the itemCode -->
<!-- and planningUom elements are required. minOccurs=0 specifies -->
<!-- that an element is not required, while nillable=true     -->
<!-- specifies that an element can appear but be empty.       -->

<xs:complexType name=itemObject>
  <xs:all>
    <xs:element name=itemCode type=scbmString minOccurs=1 maxOccurs=1/>
    <xs:element name=itemName type=scbmString minOccurs=0 maxOccurs=1
      nillable=true/>
    <xs:element name=alternateItemId type=scbmString minOccurs=0
       maxOccurs=1 nillable=true/>
    <xs:element name=description type=scbmString minOccurs=0 maxOccurs=1
       nillable=true/>
    <xs:element name=planningUom type=scbmString minOccurs=1 maxOccurs=1/>
    <xs:element name=shippingUom type=scbmString minOccurs=0 maxOccurs=1
       nillable=true/>
    <xs:element name=weight type=scbmDouble minOccurs=0 maxOccurs=1 nillable=true/>
    <xs:element name=weightUom type=scbmString minOccurs=0 maxOccurs=1
       nillable=true/>
    <xs:element name=volume type=scbmDouble minOccurs=0 maxOccurs=1 nillable=true/>
    <xs:element name=volumeUom type=scbmString minOccurs=0 maxOccurs=1
       nillable=true/>
    <xs:element name=storageRequirement type=scbmString minOccurs=0
       maxOccurs=1 nillable=true/>
  </xs:all>
</xs:complexType>

<!-- Specify that elements with the standardUomObject type can   -->
<!-- include the toUom, unitType, fromUom, and factor elements   -->
<!-- in any order. The toUom, unitType, and factor elements must  -->
<!-- appear once because minOccurs=1 and maxOccurs=1 for these   -->
<!-- elements. The fromUom element is not required. The toUom,   -->
<!-- fromUom, and factor elements have the scbmString type.     -->
<!-- The factor element has the scbmDouble type. Possible values  -->
<!-- for the toUomType element are: Weight, Volume, Length, Count, -->
<!-- and Area.                          -->
```

```
<xs:complexType name=standardUomObject>
  <xs:all>
    <xs:element name=toUom type=scbmString   minOccurs=1 maxOccurs=1/>
    <xs:element name=unitType   minOccurs=1 maxOccurs=1>
      <xs:simpleType>
        <xs:restriction base=xs:string>
          <xs:enumeration value=Weight/>
          <xs:enumeration value=Volume/>
          <xs:enumeration value=Length/>
          <xs:enumeration value=Count/>
          <xs:enumeration value=Area/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name=fromUom type=scbmString minOccurs=0 maxOccurs=1 nillable=true⇒
/>
    <xs:element name=factor type=scbmString minOccurs=1 maxOccurs=1/>
  </xs:all>
</xs:complexType>

<!-- Specify that elements with the itemUomObject type can     -->
<!-- include the itemCode, toUom, toUomType, and factor elements  -->
<!-- in any order. Each of these elements must appear once because -->
<!-- minOccurs=1 and maxOccurs=1 for these elements. The itemCode  -->
<!-- and toUom elements have the scbmString type. Possible values  -->
<!-- for the toUomType element are: Weight, Volume, Length, Count, -->
<!-- and Area. The factor element has the scbmDouble type.     -->

<xs:complexType name=itemUomObject>
  <xs:all>
    <xs:element name=itemCode type=scbmString minOccurs=1 maxOccurs=1/>
    <xs:element name=toUom type=scbmString minOccurs=1 maxOccurs=1/>
    <xs:element name=toUomType minOccurs=1 maxOccurs=1>
      <xs:simpleType>
        <xs:restriction base=xs:string>
          <xs:enumeration value=Weight/>
          <xs:enumeration value=Volume/>
          <xs:enumeration value=Length/>
          <xs:enumeration value=Count/>
          <xs:enumeration value=Area/>
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name=factor type=scbmDouble minOccurs=1 maxOccurs=1/>
  </xs:all>
</xs:complexType>

<!-- Specify that elements with the scbmString or scbmDouble type -->
<!-- can accept isNull=true or isNull=false as attributes.     -->

<xs:complexType name=scbmString>
  <xs:simpleContent>
    <xs:extension base=xs:string>
      <xs:attribute name=isNull type=simpleTrueFalse/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name=scbmDouble>
  <xs:simpleContent>
    <xs:extension base=xs:double>
      <xs:attribute name=isNull type=simpleTrueFalse/>
    </xs:extension>
  </xs:simpleContent>
```

```
</xs:complexType>

<!-- Specify that elements with the scbmDT type is restricted to the  -->
<!-- datetime yyyy-mm-ddTHH:MM:SS format with the specified pattern of -->
<!-- values.                                  -->

<xs:simpleType name=scbmDT>
  <xs:union>
    <xs:simpleType>
      <xs:restriction base=xs:dateTime/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base=xs:string>
        <xs:pattern value=[0-9][0-9][0-9][0-9]-[0-1][0-9]-[0-3][0-9]T[0-2][0-9]:
           [0-5][0-9]:[0-5][0-9]
         id=OWDateTimeFormat.pattern/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

</xs:schema>
```

Data in Supply Chain Planning XML 3.0 Format

This sample document in Supply Chain Planning XML 3.0 format is an annotated excerpt from a Base package. This XML document conforms to the structure specified by the sample XML schema definition that is included in this Implementation Guide.

```
<!-- Specify that the document uses XML version 1.0 and the UTF-8  -->
<!-- character set. (SCBM can import files that use any character  -->
<!-- set supported by the Xerces XML parser, including ISO-8859-1, -->
<!-- ASCII, EBCDIC, UTF-16, UTF-8, and Win-1252.)          -->

<?xml version=1.0 encoding=UTF-8?>

<!-- Specify an element called scbm-extract that has a version   -->
<!-- attribute value of scp 3.0                    -->
  <scbm-extract version=scp 3.0>

<!-- Specify an element called provenance with source, comment, and   -->
<!-- timestamp information. Note: This data is not currently used in  -->
<!-- SCBM, and is provided as documentation for the extract.      -->

<provenance>
  <source>EnterpriseOne Supply Chain Management</source>
  <comment>base model</comment>
  <timestamp>2003-12-05T11:22:56</timestamp>
</provenance>

<!-- Specify that the scbm-extract element has a child element   -->
<!-- called itemList.                          -->
    <itemList>

<!-- Specify that the itemList element has a child element     -->
<!-- called item. Specify the item code, name, alternate item ID,  -->
<!-- description, planningUom, shippingUom, weight, weightUom    -->
<!-- volume, volumeUom, and storageRequirement.           -->
    <item>
      <itemCode>9797700</itemCode>
      <itemName>5900_Road</itemName>
      <alternateItemId>9797700EA</alternateItemId>
      <description>Trek 5900 OCLV 110 Road Bike with Dura-Ace </description>
      <planningUom>EA</planningUom>
      <shippingUom>PL</shippingUom>
      <weight>20</weight>
      <weightUom>LB</weightUom>
      <volume>18</volume>
      <volumeUom>Cubic Feet</volumeUom>
      <storageRequirement>FINISHED GOODS</storageRequirement>
    </item>

<!-- Specify another item child element of the itemList element   -->
<!-- Specify the item code, name, alternate item ID, description  -->
<!-- planningUom, shippingUom, weight, weightUom, volume, volumeUom -->
<!-- and storageRequirement.                    -->
    <item>
      <itemCode>9797701</itemCode>
      <itemName>5900_Road_LA</itemName>
      <alternateItemId>9797701EA</alternateItemId>
      <description>Trek 5900 OCLV 110 Road Bike with Dura-Ace Lance Armstrong⇒
 Limited Edition</description>
      <planningUom>EA</planningUom>
      <shippingUom>PL</shippingUom>
      <weight>20</weight>
      <weightUom>LB</weightUom>
      <volume>18</volume>
      <volumeUom>Cubic Feet</volumeUom>
      <storageRequirement>FINISHED GOODS</storageRequirement>
    </item>

<!-- Specify another item child element of the itemList element   -->
<!-- Specify the item code, name, alternate item ID, description  -->
```

```
<!-- planningUom, shippingUom, weight, weightUom, volume, volumeUom -->
<!-- and storageRequirement.                    -->

    <item>
      <itemCode>9797702</itemCode>
      <itemName>5500_Road</itemName>
      <alternateItemId>9797702EA</alternateItemId>
      <description>Trek 5500 OCLV 120 Road Bike with Dura-Ace</description>
      <planningUom>EA</planningUom>
      <shippingUom>PL</shippingUom>
      <weight>20</weight>
      <weightUom>LB</weightUom>
      <volume>18</volume>
      <volumeUom>Cubic Feet</volumeUom>
      <storageRequirement>FINISHED GOODS</storageRequirement>
    </item>
  </itemList>

<!-- Specify that the scbm-extract element has a child element   -->
<!-- called standardUomList.                    -->

 <standardUomList>

<!-- Specify that the standardUomList element has a child element  -->
<!-- called standardUom. Specify the toUom, unitType, fromUom and  -->
<!-- factor of the standardUom.                    -->

   <standardUom>
     <toUom>KG</toUom>
     <unitType>Weight</unitType>
     <fromUom>LB</fromUom>
     <factor>0.454545454545455</factor>
   </standardUom>

<!-- Specify another standardUomList child element called      -->
<!-- standardUom. Specify the toUom, unitType, fromUom and factor  -->
<!-- of the standardUom.                    -->

   <standardUom>
     <toUom>LB</toUom>
     <unitType>Weight</unitType>
     <fromUom>LB</fromUom>
     <factor>1</factor>
   </standardUom>

<!-- Specify another standardUomList child element called      -->
<!-- standardUom. Specify the toUom, unitType, fromUom and factor  -->
<!-- of the standardUom.                    -->

   <standardUom>
     <toUom>LT</toUom>
     <unitType>Volume</unitType>
     <fromUom>ML</fromUom>
     <factor>0.001</factor>
   </standardUom>
 </standardUomList>

<!-- Specify that the scbm-extract element has a child element   -->
<!-- called itemUomList.                    -->

     <itemUomList>

<!-- Specify that the itemUomList element has a child element    -->
<!-- called itemUom. Specify the itemCode, toUom, toUomType, and  -->
```

```
<!-- factor of the itemUom.                    -->

   <itemUom>
     <itemCode>9797700</itemCode>
     <toUom>EA</toUom>
     <toUomType>Count</toUomType>
     <factor>1</factor>
   </itemUom>

<!-- Specify another itemUomList child element called itemUom.   -->
<!-- Specify the itemCode, toUom, toUomType, and factor.      -->

   <itemUom>
     <itemCode>9797700</itemCode>
     <toUom>LB</toUom>
     <toUomType>Weight</toUomType>
     <factor>25</factor>
   </itemUom>

<!-- Specify another itemUomList child element called itemUom.   -->
<!-- Specify the itemCode, toUom, toUomType, and factor.      -->

   <itemUom>
     <itemCode>9797700</itemCode>
     <toUom>PL</toUom>
     <toUomType>Count</toUomType>
     <factor>6</factor>
   </itemUom>

<!-- Specify another itemUomList child element called itemUom.   -->
<!-- Specify the itemCode, toUom, toUomType, and factor.      -->

   <itemUom>
     <itemCode>9797701</itemCode>
     <toUom>EA</toUom>
     <toUomType>Count</toUomType>
     <factor>1</factor>
   </itemUom>
 </itemUomList>
</scbm-extract>
```

See Also

EnterpriseOne Supply Chain Business Modeler 8.12.1 User Guide, "Understanding Data for Importing Into and Exporting from Supply Chain Business Modeler"

Appendix C

Understanding the Order Promising XML Format

This appendix discusses the content and format of the Order Promising XML used by the Order Promising datastore.

Understanding the Order Promising XML Format

The XML data exchanged between EnterpriseOne and EnterpriseOne Order Promising is in Extensible Markup Language (XML) format. Order Promising uses the XML version 1.0 standard that is officially recommended by the World Wide Web Consortium as of 1998. Unlike flat-file data that uses tabs or other characters as content delimiters, data in XML format uses tags to define the data.

Order Promising 8.12.1 uses an XML format called Supply Chain Planning XML 3.0 format, which has been developed for integrating EnterpriseOne supply chain products. In Supply Chain Planning XML format, data is divided into separate XML documents, or objects. Each object includes related data that must be stored and transferred together to ensure that the data is consistent and reliable. For example, the ManufacturingOperation object includes related information about operations, alternate parts, and substitution rules.

For more information about the Order Promising XML format, you can view XML schema definitions. XML schema definitions describe valid data object formats, including the elements that can appear, the order of the elements, and the valid data values in each object. .

This table indicates the locations where you can find XML schema definitions for Order Promising XML format:

Order Promising XML Schema	***Location***
Order Promising XSD Files	In Windows:*path*\scp\8.12.1\op\xsd*object* In UNIX:*path*/scp/8.12.1/op/xsd/*object*
Order Promising Schema Definition Documentation	In Windows:*path*\scp\8.12.1\op\doc\schema\main.html In UNIX:*path*/scp/8.12.1/op/doc/schema/main.html

Note. *path* is the directory where OP is installed and *object* is the type of OP data in the OP datastore.

Example: XML Schema Definition

This sample includes annotated excerpts from the ResourceAllocation schema definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!-- Root element declaration -->
    <xsd:element name="orderPromisingDataStore">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="resourceAllocationList">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="resourceAllocation" type="
                                resourceAllocationType"
                             minOccurs="0" maxOccurs="unbounded" />
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="version" type="xsd:string" fixed="0.1"
              use="required" />
        </xsd:complexType>
    </xsd:element>

    <!-- Complex type declarations -->
    <xsd:complexType name="resourceAllocationType">
        <xsd:sequence>
            <xsd:element name="allocationId" type="xsd:string" />
            <xsd:element name="customerCode" type="xsd:string" />
            <xsd:element name="itemCode" type="xsd:string" />
            <xsd:element name="startDate" type="xsd:date" />
            <xsd:element name="endDate" type="xsd:date" />
            <xsd:element name="internalContact" type="xsd:string" />
            <xsd:element name="externalContact" type="xsd:string" />
            <xsd:element name="contractInfo" type="xsd:string" />
            <xsd:element name="reservationList">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="reservation" type="capacityAllocation
                            Type" minOccurs="0"
                         maxOccurs="unbounded" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="capacityAllocationType">
        <xsd:sequence>
            <xsd:element name="partCode" type="xsd:string" />
            <xsd:element name="locationCode" type="xsd:string" />
            <xsd:element name="onlyUseReserved" type="xsd:boolean" />
            <xsd:element name="expiryTimefence" type="xsd:int" />
            <xsd:element name="expiryDay" type="DayOfWeek" />
            <xsd:element name="expiryTime" type="xsd:string" />
            <xsd:element name="unitOfMeasure" type="xsd:string" />
            <xsd:element name="weeklyReservationList">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="weeklyReservation"
                          type="capacityAllocationDetailType"
                          minOccurs="0" maxOccurs="unbounded" />
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:complexType name="capacityAllocationDetailType">
        <xsd:sequence>
            <xsd:element name="periodStartDate" type="xsd:date" />
            <xsd:element name="reservedCapacity" type="xsd:double" />
        </xsd:sequence>
    </xsd:complexType>


    <xsd:simpleType name="DayOfWeek">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="Monday"/>
            <xsd:enumeration value="Tuesday"/>
            <xsd:enumeration value="Wednesday"/>
            <xsd:enumeration value="Thursday"/>
            <xsd:enumeration value="Friday"/>
            <xsd:enumeration value="Saturday"/>
            <xsd:enumeration value="Sunday"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

You can view sample data objects in Order Promising 8.12.2 XML format. Sample data objects are saved in the *path*`\scp\8.12.1\op\data\opserver\datastore\`*object* directory in Windows and the *path* `/scp/8.12.1/op/data/opserver/datastore/`*object* directory in UNIX, where *path* is the directory where Order Promising is installed and *object* is the type of Order Promising data in the Order Promising datastore.

Appendix D

Understanding Sales Order Inquiry Error Codes

This appendix provides information about the Order Promising sales order inquiry error codes, their causes, and possible solutions.

Understanding the Sales Order Inquiry Error Codes

Order Promising generates error codes when it detects errors during a sales order inquiry. You can use the codes to diagnose the error and resubmit the sales order inquiry.

The following table describes the error codes that appear in the summary area on the Sales Order Inquiry tab:

Error Code	*Cause*	*Possible Solution*
PROXERROR	Order Promising encountered an unspecified problem while running a proximity search.	Contact Customer Support.
PROXCUSTLOCNOTFOUND	Order Promising cannot find the customer location.	Ensure that you have specified a location for the customer.
PROXSRCNOTFOUND	The Ship From location specified in the Location table was not found. This error code indicates corrupted or incomplete data in the Location table.	Ensure that a correct Ship From location exists in the Location table.
PROXMULTI	The geographic database returned several locations with the same name.	Contact Customer Support.
PROXINVALIDGSSEARCH	The data that you are trying to send to the geographic database has problems, or the results that the server returned are invalid.	Verify the data integrity before attempting a proximity search. If the data is correct and you continue to encounter this error, contact Customer Support.
PROXBADTRAVELDISTDATA	The value in the Travel Distance field is 0. Order Promising uses the Travel Distance field in the OPScenDelivery table.	Ensure that a valid value exists for the Travel Distance field in the OPScenDelivery table.

Error Code	*Cause*	*Possible Solution*
PROXCUSTNOTFOUND	The specified customer name was not found in the Customer table.	Ensure that the customer name appears in the Customer table.

The following table describes the error codes that appear in the detail area on the Sales Order Inquiry tab:

Error Code	*Cause*
OP01	An item integrity error exists. The item is not found in the data model or a list of sourcing locations could not be found for this item. It is possible that no inventory policy has been defined for this item or related sourcing locations. Order Promising cannot initiate a sales order inquiry.
OP02	The item is unavailable for either of the following reasons: • The entire quantity cannot be sourced by the location specified in the sales order line. • The fill rate is less than 100%. In cases where multiple lines are needed to fulfill and inquiry, each line will be marked with this error code if the line does not fulfill 100% of the order. **Note.** In these case of a fill rate less than 100%, this code acts as a warning. The result is valid and the order can be committed.
OP03	The quantity is an inexact multiple. This error is caused when either of the following scenarios occur: • The fill rate is 0%. • The quantity is adjusted to respect the planning multiple. The actual quantity available might exceed the quantity displayed in the sales order line. • The quantity is adjusted to respect a product substitution ratio. **Note.** This is an invalid result. Do not commit an order containing this error code.
OP04	An invalid date was supplied by an external ERP system. Order Promising cannot initiate a sales order inquiry.
OP06	An item integrity error exists in the parts list and routing for a configured parent or a configured subassembly. The item is not found in the Item master. Order Promising cannot initiate a sales order inquiry.
OP07	The customer location does not exist in the data model.
OP08	The preferred sourcing location specified in the service objective sourcing rule does not exist in the data model.
OP09	The value in the Travel Distance field is 0. When calculating lead time, Order Promising uses the Travel Distance field in the OPScenDelivery table.

Error Code	***Cause***
OP10	A proximity search error has occurred.

Index

Numerics/Symbols

A

B

C

D

E

G

I

L

M

N

O

P

R

S

T

V

W

X