ORACLE
JD EDWARDS ENTERPRISEONE

EnterpriseOne Supply Chain Planning Demand Management 9.0 Design Studio Implementation Guide

September 2008

ORACLE

EnterpriseOne Supply Chain Planning Demand Management 9.0 Design Studio Implementation Guide
SKU E1_SCPSDS-B 0908

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

License Restrictions Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

> *U.S. GOVERNMENT RIGHTS*
>
> Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Hazardous Applications Notice

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

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

About This Documentation Preface

JD Edwards EnterpriseOne implementation guides provide you with the information that you need to implement and use JD Edwards EnterpriseOne applications from Oracle.

This preface discusses:

- JD Edwards EnterpriseOne application prerequisites.
- Application fundamentals.
- Documentation updates and downloading documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common fields in implementation guides.

Note. Implementation guides document only elements, such as fields and check boxes, that require additional explanation. If an element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common fields for the section, chapter, implementation guide, or product line. Fields that are common to all JD Edwards EnterpriseOne applications are defined in this preface.

JD Edwards EnterpriseOne Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use JD Edwards EnterpriseOne applications.

You might also want to complete at least one introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using JD Edwards EnterpriseOne menus, forms, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your JD Edwards EnterpriseOne applications most effectively.

Application Fundamentals

Each application implementation guide provides implementation and processing information for your JD Edwards EnterpriseOne applications.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals implementation guide. Most product lines have a version of the application fundamentals implementation guide. The preface of each implementation guide identifies the application fundamentals implementation guides that are associated with that implementation guide.

The application fundamentals implementation guide consists of important topics that apply to many or all JD Edwards EnterpriseOne applications. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals implementation guides. They provide the starting points for fundamental implementation tasks.

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 PeopleSoft Customer Connection website. Through the Documentation section of Oracle's PeopleSoft Customer Connection, 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 PeopleSoft Customer Connection for updates to the upgrade instructions. Oracle continually posts updates as the upgrade process is refined.

See Also

Oracle's PeopleSoft Customer Connection, http://www.oracle.com/support/support_peoplesoft.html

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

The following resources are located on Oracle's PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps
Interactive Services Repository	Support, Documentation, Interactive Services Repository

Resource	Navigation
Hardware and software requirements	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Pre-Built Integrations for PeopleSoft Enterprise and JD Edwards EnterpriseOne Applications
Minimum technical requirements (MTRs)	Implement, Optimize + Upgrade; Implementation Guide; Supported Platforms
Documentation updates	Support, Documentation, Documentation Updates
Implementation guides support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Release Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes
Release value proposition	Support, Documentation, Documentation Updates, Category, Release Value Proposition
Statement of direction	Support, Documentation, Documentation Updates, Category, Statement of Direction
Troubleshooting information	Support, Troubleshooting
Upgrade documentation	Support, Documentation, Upgrade Documentation and Scripts

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

Typographical Conventions

This table contains the typographical conventions that are used in implementation guides:

Typographical Convention or Visual Cue	Description
Bold	Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
Italics	Indicates field values, emphasis, and JD Edwards 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 letter *O*.
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
`Monospace font`	Indicates a PeopleCode program or other code example.
" " (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (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.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.

Visual Cues

Implementation guides contain the following visual cues.

Notes

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

Note. Example of a note.

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

Important! Example of an important note.

Warnings

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

Warning! Example of a warning.

Cross-References

Implementation guides provide cross-references either under 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.

Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: "(FRA) Hiring an Employee"

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

See *About This Documentation*, "ISO Country and Currency Codes," ISO Country Codes.

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in implementation guides:

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in implementation guides:

- USF (U.S. Federal)
- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

See *About This Documentation*, "ISO Country and Currency Codes," ISO Currency Codes.

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 implementation guides and other Oracle reference and training materials. Please send your suggestions to your product line documentation manager at Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A. Or email us at appsdoc@us.oracle.com.

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

Common Fields Used in Implementation Guides

Address Book Number Enter a unique number that identifies the master record for the entity. An address book number can be the identifier for a customer, supplier, company, employee, applicant, participant, tenant, location, and so on. Depending on the application, the field on the form might refer to the address book number as the customer number, supplier number, or company number, employee or applicant ID, participant number, and so on.

As If Currency Code Enter the three-character code to specify the currency that you want to use to view transaction amounts. This code enables you to view the transaction amounts as if they were entered in the specified currency rather than the foreign or domestic currency that was used when the transaction was originally entered.

Batch Number Displays a number that identifies a group of transactions to be processed by the system. On entry forms, you can assign the batch number or the system can assign it through the Next Numbers program (P0002).

Batch Date Enter the date in which a batch is created. If you leave this field blank, the system supplies the system date as the batch date.

Batch Status Displays a code from user-defined code (UDC) table 98/IC that indicates the posting status of a batch. Values are:

Blank: Batch is unposted and pending approval.

A: The batch is approved for posting, has no errors and is in balance, but has not yet been posted.

D: The batch posted successfully.

E: The batch is in error. You must correct the batch before it can post.

P: The system is in the process of posting the batch. The batch is unavailable until the posting process is complete. If errors occur during the post, the batch status changes to *E*.

U: The batch is temporarily unavailable because someone is working with it, or the batch appears to be in use because a power failure occurred while the batch was open.

Branch/Plant Enter a code that identifies a separate entity as a warehouse location, job, project, work center, branch, or plant in which distribution and manufacturing activities occur. In some systems, this is called a business unit.

Business Unit Enter the alphanumeric code that identifies a separate entity within a business for which you want to track costs. In some systems, this is called a branch/plant.

Category Code Enter the code that represents a specific category code. Category codes are user-defined codes that you customize to handle the tracking and reporting requirements of your organization.

Company Enter a code that identifies a specific organization, fund, or other reporting entity. The company code must already exist in the F0010 table and must identify a reporting entity that has a complete balance sheet.

Currency Code Enter the three-character code that represents the currency of the transaction. JD Edwards EnterpriseOne provides currency codes that are recognized by the International Organization for Standardization (ISO). The system stores currency codes in the F0013 table.

Document Company Enter the company number associated with the document. This number, used in conjunction with the document number, document type, and general ledger date, uniquely identifies an original document.

If you assign next numbers by company and fiscal year, the system uses the document company to retrieve the correct next number for that company.

If two or more original documents have the same document number and document type, you can use the document company to display the document that you want.

Document Number Displays a number that identifies the original document, which can be a voucher, invoice, journal entry, or time sheet, and so on. On entry forms, you can assign the original document number or the system can assign it through the Next Numbers program.

Document Type Enter the two-character UDC, from UDC table 00/DT, that identifies the origin and purpose of the transaction, such as a voucher, invoice, journal entry, or time sheet. JD Edwards EnterpriseOne reserves these prefixes for the document types indicated:

P: Accounts payable documents.

R: Accounts receivable documents.

T: Time and pay documents.

I: Inventory documents.

O: Purchase order documents.

S: Sales order documents.

Effective Date Enter the date on which an address, item, transaction, or record becomes active. The meaning of this field differs, depending on the program. For example, the effective date can represent any of these dates:

- The date on which a change of address becomes effective.
- The date on which a lease becomes effective.

- The date on which a price becomes effective.
- The date on which the currency exchange rate becomes effective.
- The date on which a tax rate becomes effective.

Fiscal Period and **Fiscal Year**

Enter a number that identifies the general ledger period and year. For many programs, you can leave these fields blank to use the current fiscal period and year defined in the Company Names & Number program (P0010).

G/L Date (general ledger date)

Enter the date that identifies the financial period to which a transaction will be posted. The system compares the date that you enter on the transaction to the fiscal date pattern assigned to the company to retrieve the appropriate fiscal period number and year, as well as to perform date validations.

EnterpriseOne Demand Management Design Studio 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, chapter, or Implementation Guide.

Related Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

The EnterpriseOne Demand Management Design Studio Implementation Guide provides you with information about how to implement and use the EnterpriseOne Demand Management Design Studio system. Additional essential information describing deployment and supplemental third-party software options resides in the Supply Chain Planning Hardware and Software Requirements Guide. You should be familiar with the contents of this guide.

Ordering Printed Documentation

You can order printed, bound volumes of the complete Oracle documentation that is delivered on the Implementation Guides CD-ROM. Oracle makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order the documentation in the following ways:

- Electronic mail: appsdoc_us@oracle.com
- FAX: 650–506–7200 Attn: Oracle EnterpriseOne Demand Management Manager
- Postal Service:

  Oracle EnterpriseOne Demand Management Manager

  Oracle Corporation

  500 Oracle Parkway

  Redwood Shores, CA 94065

  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

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 Implementation Guides:

Typographical Convention or Visual Cue	Description
" " (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 code syntax. Options are separated by a pipe (|).
[ ] (square brackets)	Indicate optional items in code syntax.
Cross-references	Implementation Guides provide cross-references either following 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

Implementation Guides 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 Implementation Guides and other Oracle reference and training materials. Please send your suggestions to:

Oracle Sales and Operations Planning Documentation Manager

Oracle Corporation

500 Oracle Parkway

Redwood Shores, CA 94065

USA

Or email comments to: appsdoc_us@oracle.com.

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

CHAPTER 1

Getting Started with EnterpriseOne Demand Management

This chapter provides an overview of EnterpriseOne Demand Management and discusses:

- EnterpriseOne Demand Management business processes.
- EnterpriseOne Demand Management integrations.
- EnterpriseOne Demand Management implementation.

EnterpriseOne Demand Management Overview

EnterpriseOne Demand Management is a collaborative application in which forecasts from different users (called forecast versions) are compared and reconciled at different levels of abstraction and completeness. The collaborative environment is valuable when you need to calculate product demand that reveals customer patterns, market behavior, and forecasting similarities and differences among users.

The Design Studio and the Consensus Conference Room components are collectively referred to as Demand Consensus. Both of these components are connected to a database that stores model information and forecast data. The demand model contains data for forecasting and for the business model. The business model contains specific structural information about a business.

Using a centralized database, interaction between the components of EnterpriseOne Demand Management can occur seamlessly and in realtime. A variety of users, such as sales and marketing personnel, can add forecast data into the same centrally controlled forecast model, and then to compare forecast data. Other features such as exception handling and reconciliation let users check forecast data for errors and anomalies, and combine all of the forecast data into one enterprise forecast.

Design Studio

The Design Studio is a desktop workspace where you create, customize, and maintain demand model information. The Design Studio is connected to a database where model information and forecast data is stored. The demand model contains data for forecasting and for the business model. The business model contains specific structural information about a business. You can use the Design Studio to configure demand model information and allow the information to be used with other EnterpriseOne and external applications.

You can create a new model or modify an existing demand model by sorting, classifying, and assigning properties. When the information about the demand model is defined, it is then configured into an aggregated hierarchical structure that defines the relationships between the units in the hierarchy.

Consensus Conference Room

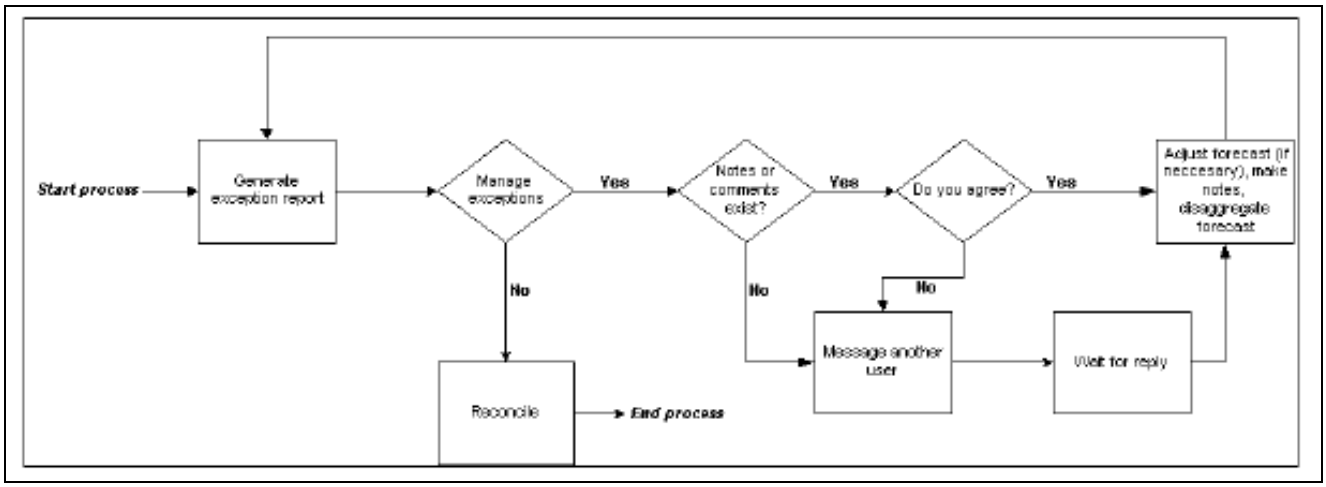
The Consensus Conference Room is a Web-based collaborative application in which you can compare and reconcile forecasts from different users (called forecast versions) at different levels of abstraction and completeness.

The Consensus Conference Room enables various users to compare forecast data for the same product, location and channel where potential exists for demand (called the demand point). When users compare multiple forecast data for the same demand point, they can correct with minimum effort any errors or miscommunications that might occur.

As you use the Consensus Conference Room, you will be able to recognize when forecast data is inaccurate or flawed. Using the collaborative forecasting process, you access data easily, identify inaccuracies, and manage these inaccuracies in forecast data across multiple business units.

EnterpriseOne Demand Management Business Processes

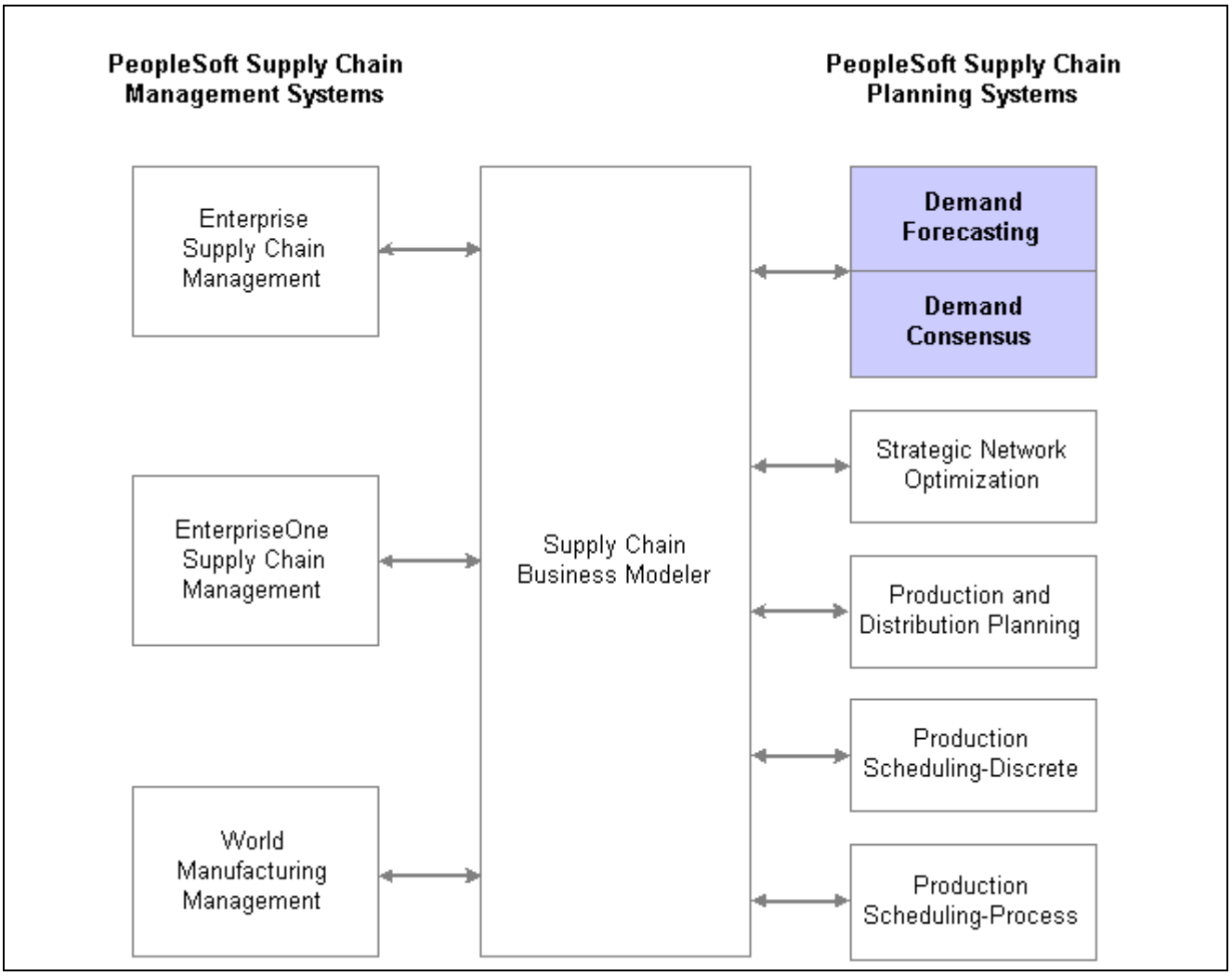
The following process flow illustrates the Demand Management process:



EnterpriseOne Demand Management business process flow

EnterpriseOne Demand Management Integrations

EnterpriseOne Demand Management can integrate with other EnterpriseOne Supply Chain Management and EnterpriseOne Supply Chain Planning systems through the EnterpriseOne Supply Chain Business Modeler application. The following graphic illustrates how EnterpriseOne Demand Management integrates with other applications through EnterpriseOne Supply Chain Business Modeler:



EnterpriseOne Demand Management integrations

Using EnterpriseOne Supply Chain Business Modeler, you can transfer supply chain data from EnterpriseOne Supply Chain Management systems into EnterpriseOne Demand Management. You can then use Demand Management to create enterprise demand forecasts and inventory safety targets based on the data.

After creating forecasts in EnterpriseOne Demand Management, you can use EnterpriseOne Supply Chain Business Modeler to transfer the plans to another EnterpriseOne Supply Chain Planning system for further refinement or to a EnterpriseOne Supply Chain Management system for further refinement or implementation.

EnterpriseOne Supply Chain Management Systems

EnterpriseOne Supply Chain Management systems such as Enterprise Supply Chain Management, EnterpriseOne Supply Chain Management, and World Manufacturing Management, provide EnterpriseOne Supply Chain Business Modeler with the supply chain data that EnterpriseOne Demand Management uses to generate accurate forecasts. The data includes information about items, branches, inventory policies, and manufacturing processes.

After EnterpriseOne Demand Management creates optimal forecasts using the data, you can transfer the forecasts through EnterpriseOne Supply Chain Business Modeler to a EnterpriseOne Supply Chain Management system for further refinement or implementation.

EnterpriseOne Sales and Operations Planning

Using EnterpriseOne Supply Chain Business Modeler, you can transfer supply chain data from EnterpriseOne Demand Management to EnterpriseOne Sales and Operations Planning.

EnterpriseOne Strategic Network Optimization

Using EnterpriseOne Supply Chain Business Modeler, you can transfer demand forecasts from EnterpriseOne Demand Management into EnterpriseOne Strategic Network Optimization. You can then use Strategic Network Optimization to create inventory build targets and sourcing recommendations based on the data.

After creating supply chain plans in EnterpriseOne Strategic Network Optimization, you can use EnterpriseOne Supply Chain Business Modeler to transfer the plans to another EnterpriseOne Supply Chain Planning system for further refinement or to a EnterpriseOne Supply Chain Management system for further refinement or implementation.

EnterpriseOne Production and Distribution Planning

Using EnterpriseOne Supply Chain Business Modeler, you can transfer supply chain plans from EnterpriseOne Demand Management into EnterpriseOne Production and Distribution Planning. You can then use EnterpriseOne Production and Distribution Planning to create net deployment requirements and net production requirements based on the data.

After creating supply chain plans in EnterpriseOne Production and Distribution Planning, you can use EnterpriseOne Supply Chain Business Modeler to transfer the plans to another EnterpriseOne Supply Chain Planning system for further refinement or to a EnterpriseOne Supply Chain Management system for further refinement or implementation.

EnterpriseOne Order Promising

Using EnterpriseOne Supply Chain Business Modeler, you can transfer supply chain data from a EnterpriseOne Supply Chain Management system to create a model in EnterpriseOne Order Promising.

EnterpriseOne Production Scheduling - Discrete and EnterpriseOne Production Scheduling: Process

Using EnterpriseOne Supply Chain Business Modeler, you can transfer supply chain plans from EnterpriseOne Demand Management into EnterpriseOne Production Scheduling - Discrete and EnterpriseOne Production Scheduling - Process. The EnterpriseOne Supply Chain Planning production scheduling applications can then produce optimal production schedules for meeting the production targets.

Supplemental information about third-party application integrations is located on the EnterpriseOne Customer Connection website.

EnterpriseOne Demand Management Implementation

The EnterpriseOne Demand Management implementation process can be divided into the following steps:

- Installing EnterpriseOne Demand Management.
- Importing data into EnterpriseOne Demand Management.
- Creating a demand model.
- Adding forecast versions and assigning permissions.

- Importing products, locations, and channels into a demand model.
- Reviewing forecast data.
- Managing exceptions.
- Disaggregating the forecast.
- Reconciling the forecast.

Installing EnterpriseOne Demand Management

This section describes how to install EnterpriseOne Demand Management.

Step	Reference
Install EnterpriseOne Demand Management.	*Demand Management Installation Guide for Windows*. *Demand Management Installation Guide for UNIX*. *Demand Management Installation Guide for LINUX*.

Importing Business Model Data into EnterpriseOne Demand Management

This section describes how to import business model data.

Step	Reference
Import data.	“Importing and Exporting Data”.

Creating a Demand Model

This section describes how to create a demand model.

Step	Reference
Create a demand model.	“Creating a Demand Model”.

Adding Forecast Versions and Assigning Permissions

This section describes how to add forecast versions and assign permissions.

Step	Reference
Add a forecast version.	“Working with Forecast Versions”, “Adding a Forecast Version.”

Importing Products, Locations, and Channels into a Demand Model

This section describes how to import products, locations, and channels into a demand model.

Step	Reference
Import products.	“Introducing New Products into Demand Points”, “Importing Products.”
Import locations.	“Introducing New Products into Demand Points”, “Importing Locations”.

Step	Reference
Import channels.	"Introducing New Products into Demand Points", "Importing Channels."
Identify new demand points.	"Introducing New Products into Demand Points", "Identifying New Demand Points."

Reviewing Forecast Data

This section describes how to review forecast data.

Step	Reference
Add your name to the review list.	"Reviewing Forecast Data", "Adding Your User Name to the Review List," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*
Email or message other Consensus Conference Room users.	"Reviewing Forecast Data", "Emailing Other Stakeholders from Within the Consensus Conference Room," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.* "Reviewing Forecast Data", "Messaging Another Consensus Conference Room User," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*

Managing Exceptions

This section describes how to manage exceptions.

Step	Reference
Create an exception report.	"Managing by Exception", "Adding an Exception Report Using the Exception Report Wizard," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*
View the exceptions.	"Managing by Exception", "View Exceptions,"*EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*

Disaggregating the Forecast

This section describes how to disaggregate the forecast.

Step	Reference
Set the disaggregation options.	"Working with Forecast Data in the Workshop", "Setting the Options for Disaggregation," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*
Disaggregate the forecast.	"Working with Forecast Data in the Workshop", "Disaggregating Forecast Data," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*

Reconciling the Forecast

This section describes how to reconcile the forecast.

Step	Reference
Set the parameters to reconcile the forecast.	"Reconciliation", "Setting Up Reconciliation Parameters,"*EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*
Generate a reconciled forecast.	"Reconciliation", "Generating a Reconciled Forecast," *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide.*

CHAPTER 2

Understanding the Design Studio

This chapter provides an overview of the Design Studio and discusses:

- Understanding model structure.
- Working within an aggregation hierarchy.
- Working with forecast data in the demand model.

Design Studio

The Design Studio is a workspace where you create, customize, and maintain demand model information. The Design Studio is connected to a database where model information and forecast data is stored. The demand model contains data for forecasting and for the business model. The business model contains specific structural information about a business. You can use the Design Studio to configure demand model information and allow the information to be used with other EnterpriseOne and external applications.

You can create a new model or modify an existing demand model by sorting, classifying, and assigning properties. When the information about the demand model is defined, it is then configured into an aggregated hierarchical structure that defines the relationships between the units in the hierarchy.

Model Structure

Different functional areas in an organization - for example, sales and marketing - often vary on how they model their enterprise information. These variations in structure, size, and complexity make cross-functional comparisons difficult. Creating a model that the entire enterprise can use simplifies the process, and saves money and time.

A model consists of two parts:

- Demand model.
- Business model.

Demand Model

The demand model contains the business model, forecast version information, and forecast data. Forecast versions use business model structure, which provides containers for forecast data that is eventually viewed by users of the Design Studio and the Forecast Studio. The forecast versions also have users assigned to them. The administrator grants forecast version permissions to read, write, or reconcile forecast data to users of the demand model. Setting permissions and structure is important framework for viewing and working with forecast data in other applications.

Elements of the demand model include:

- Business model.
- Forecast data.
- Forecast versions.
- Historical data (sales and forecast).

You can import a demand model from an outside source, but only those fields specified in the demand model will be imported. Other fields will not be imported.

Business Model

The business model contains the structure of the model and the demand model uses the business model structure to hold the forecast data. The business model contains many different levels of information at different levels of detail. The most detail appears in the lower levels while broader levels of detail appear in the upper levels.

Configuring the data dependencies within the business model creates the hierarchical structure that resembles and functions like a pyramid. When viewing the information in the business model, subordinate (child) elements appear under primary (parent) elements. You can expand the view to display all of the properties under the product, location, or channel; or to hide all of the child properties. Parent-child relationships form the basis of the hierarchical structure.

Elements of the business model include:

- Product, location and channel.
- Product, location and channel property names, and property values.
- Demand points.
- Effective dates.
- Unit of measure.
- Forecast history horizon.
- Sales horizon.
- Aggregation hierarchy.

Product, Location and Channel

A business model defines the environment within which the business uses and maintains product, location, and channel information. These three elements are arranged in a three-dimensional structure that resembles a pyramid. To view the relationships between products, locations, and channels you must assign a demand point before you save the model. These demand points form the three-dimensional structure for the model.

Products represent a good or service. Locations represent a place where products are shipped or sold. Channels represent a set of companies, distributors, or individuals who participate in the movement of goods. The combinations of product, location, and channel define a three-dimensional business model. Any combination of product, location, and channel with potential demand for a product is called a demand point. Some combinations of product, location, and channel have no demand point because not all products are shipped or sold at all locations; therefore, no forecast data is necessary or available.

Product, Location and Channel Property Names, and Property Values

Products, locations, and channels each have distinct properties that are used to describe them. Properties are name-value pairs. A property name describes what type of information is stored in a property - for example, region, country, or continent. All channels share the same property names. All locations share the same property names. All products share the same property names.

However, individual products (locations and channels) have a specific property value that is associated with each property name; for example, different locations might have the property values Boston, Denver, or Toronto associated with the property name city. Property names group data into levels while the property values associated with that name define the groups on that level. At the region level, groups can exist for Boston, Denver, and Toronto.

Property Level	Property Name	Property Value
Location	ID	L2
	region	Boston, Denver, Toronto
	country	USA, Canada
	continent	North America

Demand Points

A demand point is a point in the business model where potential exists for a product to be sold from a location into a channel. This point is a combination of product, location, and channel; and is useful when determining if a company can sell products at a particular location and channel. For example, mountain bikes (product) sold in Boston (location) from a retailer in Mass Marketing (channel) represents a valid demand point.

Demand points at the lowest level in the model, called *leaves*, are combinations of individual product, locations, and channels. Demand points that are higher in the business model hierarchy represent user defined aggregations or sets of products, locations, and channels - such as a business unit in a geographical region. In the Design Studio, the user specifies all of the leaf demand points. The value of the demand points for levels that are higher in the model are automatically created, based on the existence of points below them. Demand points, both aggregated and non-aggregated, are identified by their product path, location path, and channel path.

If you add a product, location, or channel, you must assign a demand point before you save the model so that you can view this information in the Consensus Conference Room. If a demand point is not assigned and the model is saved, then the product, location, and channel information cannot be viewed in the Consensus Conference Room.

Demand points also have effective dates assigned to them. This identifies the period in which the demand point is available. Demand points quantities can be “fixed” in your model for a specific aggregation hierarchy, protected from changes during aggregation and disaggregation.

Effective Dates

Effective dates indicate when demand points are available or unavailable in the forecasting process. Effective dates are set and changed in the demand model using the Design Studio. You can set the effective start date and end date for a single demand point or for multiple demand points.

Setting effective dates for demand points benefits the forecasting process in these instances:

- Products are not available at a location during a specific date period.
- Channels do not carry a product during a specific date period.
- Product lines are discontinued or reintroduced on a specific date.

Knowing the effective start and end dates helps to assure demand planners that forecasting data is valid. To avoid generating demand point exceptions in the forecasting process, demand planners should apply effective dates to demand points.

Unit of Measure

The business model accommodates unit of measure. Depending on how you want to view data, you can specify which unit of measure to use. You can set up a conversion table that enables you to create a custom unit of measure conversion table that maps each unit of measure that is used in the business model to a specific value. The table converts units that use the same system of measure - for example, grams and kilograms - and those that are disparate, such as dollars and pallets.

For example, one Pallet is equivalent to 20 times the base unit. One Each is equivalent to fives times the base unit. One Each is four times the value of one Pallet, and one Palette is .25 the value of one Each.

Note. Any change to the base unit of measure overrides the conversion rates for the demand point, and resets the conversion rates to 1. You also lose the dynamic unit of measure information when you change the base unit of measure. Therefore if the base unit of measure value changes, the value of each may not always be 5 dollars.

Name	Conversion Rate	Conversion Value
US Dollar (base unit)	1 dollars	1
Each	5 dollars	0.02
Pallets	20 dollars	0.05

Dynamic Unit of Measure

Product prices can vary over time due to such factors as promotions, fluctuations in exchange rates, and so on. To keep up with these changes, you can dynamically set prices and conversion rates for products at specific demand points. Dynamically setting units of measure like conversion rates and prices enables you to:

- Trace the effects of price changes on demand for a product.
- Set one or more conversion rates for products at specified periods in the horizon of a demand model.

When working with dynamic unit of measure data, you would typically generate a forecast for a particular demand point, for example in dollars. Then, to see the forecast in other units of measure, you recreate the original forecast, but this time set the unit of measure to pallets. You do this by applying the conversion rate on a bucket per bucket basis, while respecting the effective dates on which the conversion rate is active.

Note. Dynamic unit of measure information, including price, is not stored as a time series.

Forecast Horizon

Included in the business model is the forecast horizon. The forecast horizon is the time period for which users generate forecasts. It is specified by the first date for which you are forecasting (the start date), the size or granularity of the time buckets into which the demand is broken down, and the number of time buckets being forecasted from the start date. Setting the horizon values for the business model enables you to compare forecast data for specific time periods that are measured in specific time buckets.

A horizon is defined by: the date when the horizon starts; the size of the time buckets that the horizon is broken up into:

- Daily
- Monthly
- Weekly
- Yearly
- Four weekly
- 445
- 454
- 544

The horizon is also defined by the number of buckets from the start date to the end date that the horizon covers. You can calculate the horizon from the start date of each time bucket and the end date of the horizon. For example, a horizon starting on January 1, 2003, with five weekly time buckets has time buckets starting on the following dates: January 1, January 8, January 15, January 22, and January 29. The entire horizon ends on February 4, 2003.

See Also

Sales History Horizon

The sales history horizon must end exactly one day before the forecast horizon begins. Both horizons must use the same time bucket size; however, they do not need to have the same number of time buckets. Using the previous example, a valid sales history horizon might start on December 4, 2002, and contain four weekly time buckets.

The Design Studio automatically performs all of the date calculations for you. You must use at least one time bucket in both horizons for calculation to occur. Even if you don't have any historical sales data, you must have a sales history horizon with at least one time bucket and no data in the time bucket.

Note. The forecast history has the same horizon as the sales history.

See Also

Aggregating Information

Aggregating information enables you to see the total forecast or sales history for a set of demand points in a demand model. Aggregation automatically calculates the value for a demand point by summarizing the associated values from the demand points in the aggregation level immediately below. Navigation and roll-up operations move down and up through the aggregation hierarchy. You can access more levels of detail through navigation. Roll-up operations display a summarized level of information. You can choose to make one aggregation hierarchy in your demand model “fixable”, so that values in different levels can be fixed, protecting them from change during the aggregation and disaggregation processes.

See Also

Chapter 2, "Understanding the Design Studio," Aggregating Information, page 14

Working Within an Aggregation Hierarchy

One of the challenges in forecasting is the ability to view forecast data that provides you with the perspective that is most meaningful to you. The ability for each user to create an aggregation hierarchy with demand point data that is meaningful to him or her will undoubtedly

- Increase usability.
- Improve reporting.
- Provide the best forecasting results.

Multiple aggregation hierarchies enable you to create several different views of the entire demand model including only those demand points that you want included in forecast data. For example, when a model is created, forecast data that the marketing department is responsible for is configured differently than that of the sales department. Marketing is interested in data at higher levels of abstraction while the sales department wants to configure forecast data at a more granular level. You can configure aggregation hierarchy to suit forecasting needs, customizing the view of the forecast data from the base demand model.

User permissions for aggregation hierarchies are set in the User Manager. The demand point data at the lowest-leaf level are the same for each aggregation hierarchy. When switching between aggregation hierarchies, the aggregation levels and demand points that display in the aggregation display different perspectives in the data. Only one aggregation hierarchy can be denoted as the fixing hierarchy at a time.

Demand point data that displays in each aggregation hierarchy depends upon these parameters:

- Security permissions.
- Exception reports.
- Demand point sets.
- Scenarios configuration.

Aggregation Wizard

The Aggregation Wizard is a tool that enables you to specify the aggregation hierarchy for the business model. You can add, edit, rename, and delete aggregation hierarchies.

The Aggregation Wizard uses this series of steps to help build the aggregation hierarchy:

- Define the name of the aggregation hierarchy.
- Create aggregation levels, as follows:
  - Select and arrange the order of the properties for product levels.
  - Select and arrange the order of the properties for location levels.
  - Select and arrange the order of the properties for channel levels.
  - Define the aggregation levels into an aggregation hierarchy.

After you define the aggregation hierarchy name, you define the product, location, and channel levels. You define an aggregation level in the aggregation hierarchy by choosing one each of the previously defined product levels, location levels, and channel levels.

For each aggregation level, you can review and manipulate the aggregated demand for the cross-product of all of the values on the chosen product, location, and channel properties. The wizard provides you with lists from which you can choose properties when you create the aggregation hierarchy. You need to change at least one property to add another level to the aggregation hierarchy. You can use the same properties in one or two of the lists when creating the aggregation hierarchy.

To create an aggregation hierarchy, you create a nested set of the aggregation levels. Using this type of hierarchy, you can refer to entire sets of products, locations, or channels in the same way that you would refer to a single product, location, or channel. For example, assume that the channel level is *top*, the location level is*region*, and the product level is *product family*. In this example, you would see the aggregated demand for each product family, at each region, summarized by all channels.

Aggregation Wizard

The Aggregation Wizard helps to build the aggregation hierarchy by using a series of steps. Steps one to three select and arrange the order of the properties for products, locations, and channels levels. When all of the properties are selected, you arrange them into an aggregation hierarchy.

Step four provides three lists of options - one for each product, location, and channel - that enable you to select the properties to create an aggregation level. Each time that you select a property from the product, location, and channel lists, you must click the next button to add the properties to the corresponding product, location, and channel aggregation display box.

To add a new aggregation level hierarchy, continue to add combinations of product, location, and channel properties. You can use the same level in one or two of the options lists when creating the aggregation hierarchy. You need to change at least one property to add another level to the aggregation hierarchy.

Note. The *name* property for the product, location, and product levels appears as the lowest level in the aggregation. The system creates this level automatically for you.

The highest aggregation level for each product, location, and channel level is Top/Top/Top while the lowest level of aggregation is Name/Name/Name:

Level	Product	Location	Channel
1	Top	Top	Top
2	Business Unit	Country	Name
3	Product Group	Region	Name
4	Name	Name	Name

Each aggregation level represents a layer of grouping by product, location, and channel levels. By using the information in the previous table, each level is broken down to display property names and associated property values. For example, by using the channel Mass Marketing, all of the associated property names and property values appear:

- Level 1 represents the highest level of aggregation for all products, locations, and channels.

  You can view the total demand for each product, location, and channel:

Product	Location	Channel			
Property	Value	Property	Value	Property	Value
Top	All	Top	All	Top	All

- Level 2 represents the aggregated demand for products at the business unit level Recreational, at the channel Mass Marketing, and at the location United States:

Product	Location	Channel			
Property	Value	Property	Value	Property	Value
Business Unit	Recreational	Country	United States	Name	Mass Marketing

- Level 3 represents the aggregated demand for:
- Mountain Bike products at Mass Marketing, Boston.
- Touring Bike products at Mass Marketing, Boston.

Product	Location	Channel			
Property	Value	Property	Value	Property	Value
Product Group	Bikes	Region	Boston	Name	Mass Marketing

- Level 4 represents the lowest level of aggregation for the product name, location name, and channel name.

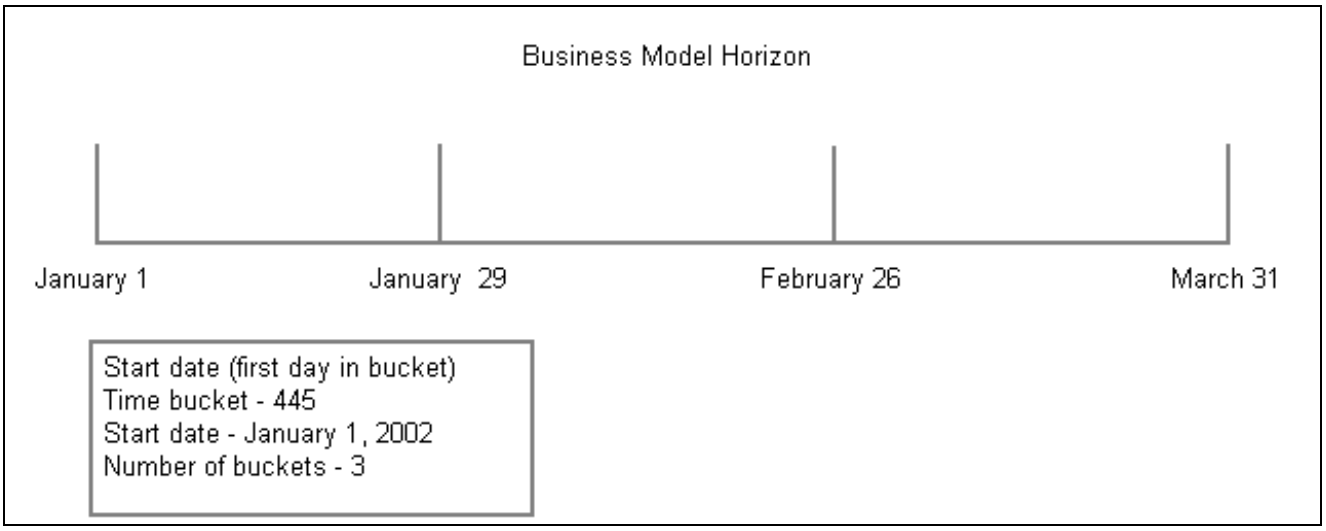
Once the aggregation levels are defined, you can rename them to meaningful names. For example Level 1, might have a default value called Top/Top/Top. You can rename this level to something that is meaningful to the business.

Working With Forecast Data in the Demand Model

The demand model holds the structure for forecast data. Understanding how forecast data behaves within the model structure helps you to understand potential behavior and results from the data.

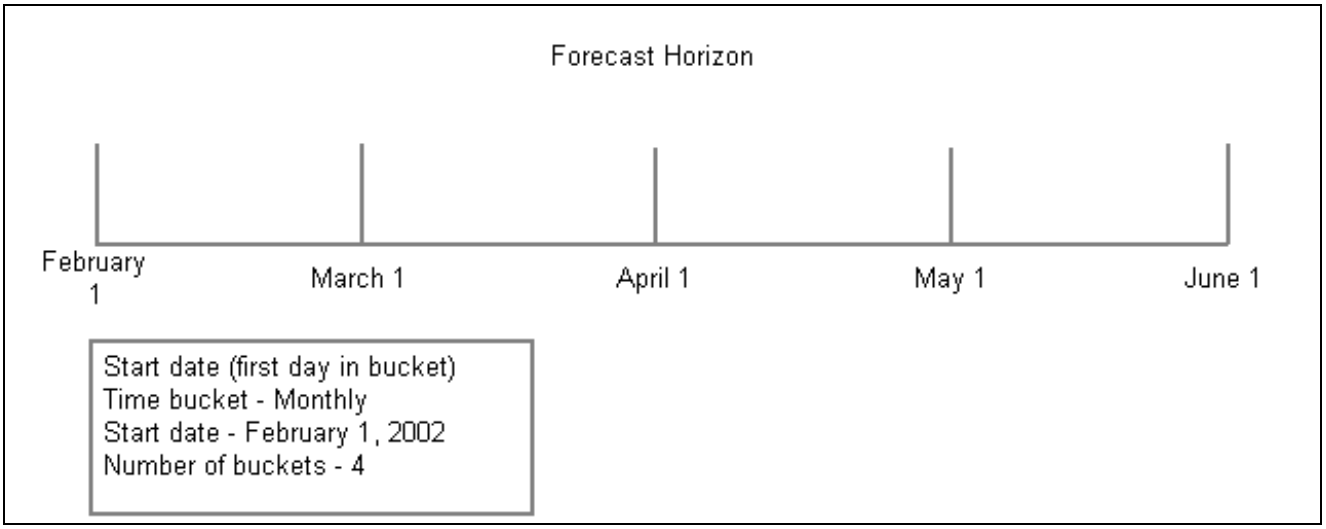
Business Models, Horizon Settings and Forecast Data

Consider the business model forecast horizon to be:



Sample business model forecast horizon

And the horizon for the forecast that you are importing is:



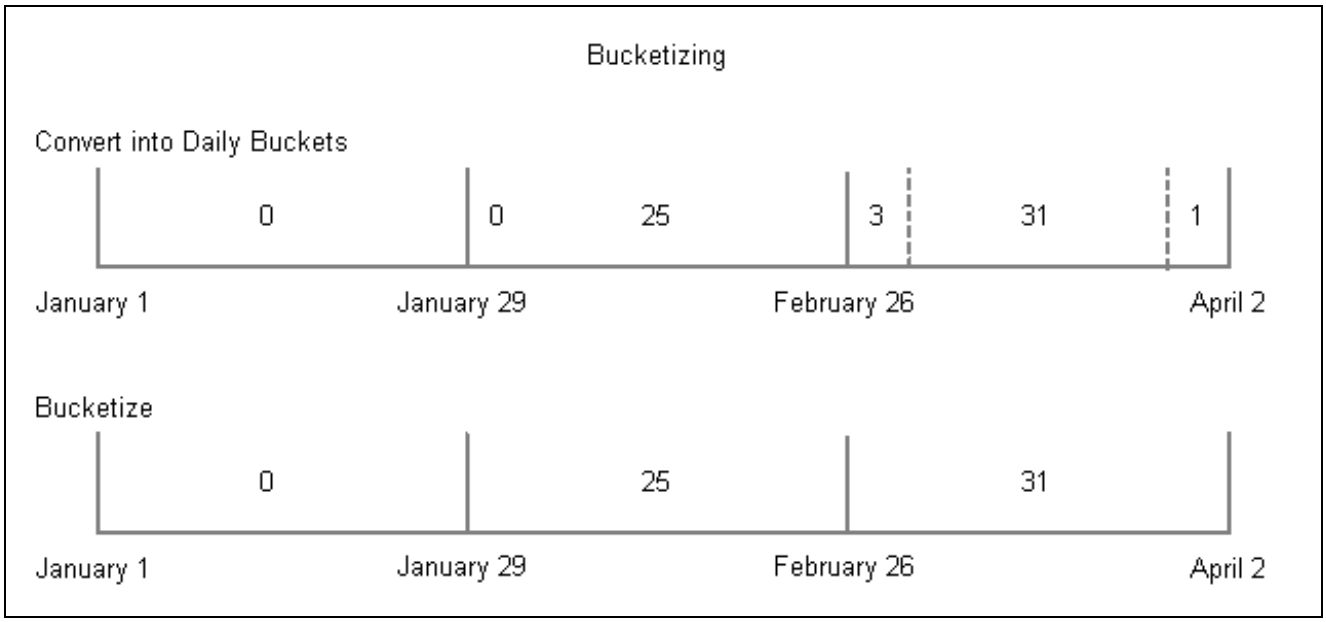
Sample imported forecast horizon

The forecast's horizon is different from the horizon of the business model. Assume that demand in each time bucket of the horizon is 28, 31, 30, and 31, respectively. To import the forecast with this horizon into the business model, it must be re-bucketized.

Bucketizing distributes the demand from one time period into another, providing another view of the data. The demand that occurs in the forecast's horizon is redistributed into the structure of the horizon of the business model. Dates are different, bucket sizes change, and days that do not fit into one bucket are shifted into the next bucket.

In this example, the forecast horizon starts on February 1 and runs to May 31. The business model horizon starts on January 1 and runs to March 31. Any data that falls outside of the business model horizon is not recognized; therefore, the forecasts for April 1 through May 31 are discarded. No forecasts exist for the month of January, so the forecast for the first time bucket in the business model, January 1 through January 28, is empty.

The forecast horizon uses monthly buckets, but the business model horizon uses a 445-bucket, four-week, four-week, and five-week cycles. This bucket format is useful when providing forecast information on a quarterly basis. The demand of the forecast model is first broken down into daily time buckets where each day gets a fair share of the monthly demand (1/28 for February, 1/31 for March, and so on). Each daily demand is then assigned to the correct bucket of the 445 time buckets in the business model. These values are then aggregated together to calculate the total for each of the business model time buckets. This diagram illustrates the conversion:



The aggregation process

Note that when bucketizing occurs, not all original forecast values can fit within the new horizon. In addition, data can be aggregated or moved to new buckets. Therefore, you might lose information: After you have configured the business model horizon and want to export it back to the originating system, if the horizon was changed during the bucketizing process, this new data overwrites the data in the originating system. For example, supposed that the data in the originating system is measured in monthly time buckets and the business model is configured in yearly time buckets. When the business model is exported to the originating system, it remains in yearly time buckets and can overwrite the existing data.

Forecast and History Horizon Guidelines

Once you have set the horizons for the business model, all demand, including forecasts and historical demand, is stored internally in the calculated time buckets. If you import forecast or sales history data by using the same horizon as the business model, the data is stored internally exactly as it is imported. However, you can import forecasts with a horizon that is totally different from the business model. In such a case, the forecast values are re-bucketized and fit into the business model horizon structure.

These guidelines explain how the system processes horizon data:

- If any of the forecast data falls outside the horizon's start and end dates, the data outside the horizon is discarded.
- If the data is at a more granular level than data in the time buckets of the business, the values are aggregated into the buckets of the business.

  For example, suppose that you are importing daily data into a business model with weekly time buckets. In this case, the daily demand is aggregated into weekly demand.

- If the data is at a larger granular level than data in the business model, it is broken down by using a fair-share algorithm.

  For example, supposed that you import weekly data into a business model with daily time buckets. In this case, each day gets 1/7 of the weekly demand.

In all cases, if the imported data crosses time bucket boundaries, it is apportioned to the correct time buckets within the business model.

Disaggregation

Disaggregation enables a user to compare forecast data, which was forecasted at a higher level, with another user's data that was forecasted at a lower level of abstraction. The process of disaggregation takes edited values at a demand point (non-leaf) and distributes forecast data at the levels (leaves) below it so that they sum up to the original edited value. It is also used to view high level forecast data at a more granular level.

Using the business model structure, disaggregating data enables you to view and compare forecast data that is distributed below your level of abstraction. You must set this feature to disaggregate forecast data in the Consensus Conference Room and Forecast Studio.

You can set one of the forecast versions as the disaggregation profile, which enables you to use the forecast version, in the Consensus Conference Room or Forecast Studio, as a baseline for comparing forecast data to other forecast data for specific demand points.

Disaggregating using the forecast data that is contained in a forecast version involves redistributing values to below the demand point from where you are currently disaggregating. Changing a demand point value in a forecast version allows the values beneath the change to disaggregate, based on the existing values of the selected forecast version. After disaggregation occurs, the values above the change remain the same; but the values in the lower level are changed.

CHAPTER 3

Working with the Design Studio

This chapter discusses how to:

- Start and log into the Design Studio.
- Change information in a demand model.

Starting and Logging Into the Design Studio

You start the Design Studio from the Windows Start menu. In UNIX, you start the Design Studio from the command line prompt. On both operating systems, a login dialog box appears when the application is launched. To log in to the Design Studio, you must have administrator privileges. The system administrator can assign administrator privileges to any user by changing the profile of the user from within the User Manager.

To start and log into the Design Studio:

1. Do one of the following:

   From the Windows Start menu, select Programs, EnterpriseOne Supply Chain Planning 9.0, Demand Management, Design Studio.

   At the UNIX command line prompt, type this command:

   path`/common/start/run_dm_design_studio`

   Where *path* is the directory path (for example, */opt/scp/9.0*) where the application is installed.
2. Complete the User ID and *Password* fields and then click Login.

Changing Information in a Model

Changes made to the business model can affect the structural integrity of the demand model in the database, which is particularly important when users are connected to one of the other applications in the suite and are accessing the demand model.

You should be aware that changes that you make to the demand model or business model can negatively affect other applications in the suite. For example, if you delete or change a product from the model, the system deletes the associated forecast data for that product. If you make changes to a model and do not close the Design Studio, users of one of the other applications will not be able to access the model or its associated data.

Creating a New Demand Model

The demand model contains business model structure and forecast data. After you create the demand model, you can configure the business model by adding information about product, location, channel data, horizon data, unit of measure data, and aggregation hierarchy data.

To create a new demand model, select New Demand Model from the File menu.

Opening an Existing Demand Model

You can open an existing demand model that resides in the database and edit the business model information. The demand model contains specific values for forecast data, such as values for demand points at particular levels of aggregation. It also contains business model information, such as product, location, and channel information; horizon details; unit of measure information; effective dates and aggregation hierarchy structure.

To open an existing demand model from the database:

1. Select Open Demand Model from the File menu.
2. Select the demand model that you want to open from the Demand Models list.
3. Click OK.

Deleting a Demand Model

When you delete a demand model, the system removes forecast data for this model in the Design Studio, Forecast Studio, and the Consensus Conference Room. Before you delete a demand model, you should ensure that no other users require this information. You must have administrative permissions to be able to delete demand models.

To delete a demand model:

1. Select Delete Demand Model from the File menu.
2. Select a demand model from the Demand Models list.
3. Select Delete.
4. Select Yes.

CHAPTER 4

Importing and Exporting Data

This chapter provides an overview of importing and exporting data and discusses how to:

- Export and import business model profiles.
- Export and import an aggregation hierarchy.

Understanding the Import and Export of Data

You can work with model data outside of the Design Studio. After you have made changes to the demand model, you can re-import the business model data into the database, where it will be available to other users. You do so by working with XML files. Working with XML files provides more flexibility to make changes at your own pace, without concern about changing data that is shared by other users who are using the same model in the Forecast Studio or the Consensus Conference Room.

Import and Export Model Data

The imported demand model will overwrite all of the data, such as forecast data and scenario data (in the Forecast Studio), in the existing demand model. Any changes made to the demand model, affects the structural integrity of the demand model in the database.

You can export existing model structure from the database to an XML file. The file is stored in a directory outside of the database, where you can make changes by editing the XML file.

Import and Export of Aggregation Hierarchies

An aggregation hierarchy defines different hierarchical levels for demand points and aggregated demand points. You use the aggregation hierarchy to determine aggregated data for aggregated demand points positioned in the next lower-level aggregation. You can also use the aggregation hierarchy to disaggregate a forecast from a demand point into demand points in the level below.

An aggregation hierarchy provides you with a view of the business model according to how you arrange the details. You can import an aggregation hierarchy into the Design Studio from an XML file. This feature provides users with the flexibility to work outside of the Design Studio and then import an aggregation hierarchy into the Design Studio. If you import an aggregation hierarchy into an existing demand model, you will lose any of the changes that you made in the demand model.

You can also export an aggregation hierarchy from the Design Studio to an XML file, which you can edit and store in a directory outside of the database.

Importing and Exporting Business Model Profiles

This section discusses how to:

- Import business model profiles.
- Export business model profiles.

Windows Used to Import Business Model Profiles

Form Name	Navigation	Usage
Design Studio	File, Import Business Model Profile	Opens the Import Business Model Profile dialog box.
Design Studio	File, Export Business Model Profile	Opens the Export Business Model Profile dialog box.

Importing a Business Model Profile

Access the Import Business Model Profile dialog box.

To import a business model profile:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click OK.

Note. After importing a business model profile you must reconfigure the aggregation hierarchy. For example the new business model profile may not contain the same properties used in the previous aggregation hierarchy.

Exporting Business Model Profiles

Access the Export Business Model Profile dialog box.

To export an existing business model profile:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click OK.

Importing and Exporting an Aggregation Hierarchy

This section discusses how to:

- Import an aggregation hierarchy.
- Export an aggregation hierarchy.

Windows Used to Import and Export an Aggregation Hierarchy

Form Name	Navigation	Usage
Design Studio	File, Import Aggregation Hierarchy	Opens the Import Aggregation Hierarchy dialog box.
Design Studio	File, Export Aggregation Hierarchy	Opens the Export Aggregation Hierarchy dialog box.

Importing Aggregation Hierarchies

Access the Import Aggregation Hierarchy dialog box.

To import an aggregation hierarchy:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click OK.

Exporting Aggregation Hierarchies

Access the Export Aggregation Hierarchy dialog box.

To export an aggregation hierarchy:

1. Complete these fields:
   - Save in
   - File name
   - Save as type
2. Click Save.

CHAPTER 5

Working with Forecast Versions

This chapter provides an overview of forecast versions and discusses how to:

- Add a forecast version.
- Assign forecast version permissions.
- Delete forecast versions.
- Set a forecast version as the disaggregation profile.

Understanding Forecast Versions

A user (or group of users) of Design Studio owns each forecast version. A forecast version is associated with a specific demand model and is a placeholder for forecast data. The demand model is available in the Consensus Conference Room and the Forecast Studio. Each forecast version has specific permissions assigned allowing users to read, write, or reconcile the forecast data that is contained in the forecast version.

You can add, edit, and delete forecast versions. You can also assign permissions to each forecast version. Assigning permissions to users allows them to read, write, or reconcile the forecast data that is contained in the forecast version.

For example, suppose that the Sales forecast version is owned by Fred of the Sales Department. Fred does not want Lin, the owner of the Marketing Department forecast version, to read, write, or reconcile his forecast version. Therefore, he asks the administrator to set the permissions for the Marketing forecast version so that Lin does not have read, write, or reconcile permissions.

Forecast Version Permission Assignments

You can assign permissions to forecast versions. Assigning permissions gives each user read, write, or reconcile privileges to the current forecast version. You can assign a single permission option or all of the permissions options.

For example, if you want the Sales forecast version to be read only by users Susan and Mike, you click the read option beside their names. This action ensures that Susan and Mike cannot edit or reconcile the Sales forecast version in the Consensus Conference Room or Forecast Studio.

When the demand model is saved to the database, the forecast versions and their associated permissions are available in the Consensus Conference Room and Forecast Studio.

Forecast Versions as Disaggregation Profiles

Disaggregating data enables you to compare forecast data that is below the selected forecast version's level of abstraction. You can set one of the forecast versions as the disaggregation profile.

In the demand model, every forecast version contains forecast data for particular demand points in the aggregation hierarchy. The process of disaggregation takes edited values at a demand point (non-leaf) and distributes forecast data at the levels (leaves) below it so that they sum up to the edited value.

Disaggregating using the forecast data in a forecast version redistributes values to below the demand point from where you are currently disaggregating. Changing a demand point value in a forecast version disaggregates the values beneath the change, based on the existing values of the selected forecast version. After disaggregation, the values above the change remain the same.

Adding Forecast Versions

This section discusses how to:

- Add a forecast version.
- Sort the forecast version list.
- Rename a forecast version.

Window Used to Add a Forecast Version

Form Name	Navigation	Usage
Design Studio	Edit, Forecast Versions	Displays the Forecast Versions dialog box.

Adding Forecast Versions

Access the Forecast Versions dialog box.

To add a forecast version:

1. Click Add.
2. Enter the name of the new forecast version.
3. Click OK.

Note. To add permissions for this forecast version click the Permission button. Alternatively you can add and edit forecast versions using DASH commands or the User Manager.

Do not use these characters as part of the forecast version name:

" (quote)

/ (slash)

+ (plus sign)

Using these characters can cause display errors in the Forecast Studio or the Consensus Conference Room.

Sorting the Forecast Version List

Access the Forecast Versions dialog box.

To sort the forecast version list, click the Forecast Version tab to sort the order of the list.

The list is sorted alphabetically in descending order.

Renaming Forecast Versions

Access the Forecast Versions dialog box.

To rename a forecast version:

1. Select the forecast version that you want to rename.
2. Click Rename.
3. Type a new name for the forecast version, and then press Enter.

Assigning Forecast Version Permissions

Access the Forecast Versions dialog box.

To assign forecast version permissions:

1. Select the forecast version for which you want to add permissions, and then click Permissions.
2. Select one of the options next to the Users column:
   - Read
   - Write
   - Reconcile
3. Optionally, select:
   - Select All
   - Clear All
4. Click OK.

Deleting Forecast Versions

Access the Forecast Version dialog box.

To delete forecast versions:

1. Select the forecast version that you want to delete.
2. Click Delete.
3. Click Yes.

Setting a Forecast Version as the Disaggregation Profile

Access the Forecast Version dialog box.

To set a forecast version as the disaggregation profile:

1. Select the forecast version that you want to set as the Disaggregation profile.
2. Click Set as Disaggregation Profile.

CHAPTER 6

Introducing New Products

This chapter provides an overview of new product introduction and discusses how to:

- Import products, channels, and locations.
- Import and export demand points.
- Identify new demand points.
- Clear new flags.

Understanding New Product Introduction

The product lines that you deal with are rarely static. You typically add new products to the line or phase out older products as they reach the end of their life cycles. In order to adapt the forecasts to these changes, you must be able to add or remove products and demand points from demand models.

Design Studio simplifies the process of adding and removing new products from demand models. Instead of manually entering information for new products, locations, or channels, you can import the information into Design Studio from an XML file. This XML file typically comes from an external data source, such as an ERP system, a project data management system, or the EnterpriseOne Supply Chain Business Modeler application.

The steps required to introduce new products are as follows:

1. Import new products, locations, or channels into the Design Studio from an XML file.
2. Identify new demand points in the Design Studio.
3. Filter the new products, locations, or channels in the Forecast Studio.

Note. You can also introduce new products into demand models using DASH commands.

Import Products, Locations, and Channels

Manually creating products, locations, and channels, in Design Studio can be a time-consuming task, especially if you are creating a large number of coordinates. A more efficient way to set up new products, locations, and channels is to import an XML file into the database. This XML file typically comes from an external data source, such as an ERP system, a project data management system, or the EnterpriseOne Supply Chain Business Modeler application.

You can add products, locations, and channels in two ways:

- Import sets of products, locations, and channels from the File menu.
- Import sets of products, locations, and channels using the appropriate DASH command.

See Also

Chapter 17, "Using the Demand Automation Shell," Business Model Management, page 120

Import and Export of Demand Points

A demand point is a point in the business model where potential exists for a product to be sold from a location into a channel. This point is a combination of product, location, and channel, and is useful when determining if a company can sell products at a particular location and channel. For example, mountain bikes (product) sold in Boston (location) from a retailer in Mass Marketing (channel) represents a valid demand point.

You can import demand points into the Design Studio from an XML file using either a command from the File menu or the DASH command. The source of the demand point information is usually an external data source such as an ERP or product data management system.

You can also export demand points from the Design Studio to an XML file, which you can edit and store in a directory outside of the database.

Identify New Demand Points

After you have imported a set of demand points into Design Studio, you can indicate that these demand points are new and require attention - for example, you might need to associate a sales history with the demand point. You identify new demand points by setting a status flag for the demand point. When you view a demand point, the flag appears in the Status column of the Demand Point pane.

Normally, demand points are marked as new when they are imported into Design Studio. However, in Design Studio you can also mark existing demand points as new in order to work with their histories in the Forecast Studio.

After you have added demand points to a demand model, you can:

- Add a new conversion rate.
- Change the base unit of measure for the demand point.
- Assign the demand point start and end dates.
- Change the forecasting horizon for the demand point.

  Changing the horizon for the demand point also changes it for the entire demand model.
- Set or clear the New flag for any demand point.

Importing Products, Locations, Channels, and Demand Points

This section discusses how to:

- Import products.
- Import locations.
- Import channels.
- Import demand points.

Windows Used to Import Products, Locations, Channels, and Demand Points

Form Name	Navigation	Usage
Design Studio	File, Import Products	Displays the Import Products dialog box.
Design Studio	File, Import Locations	Displays the Import Locations dialog box.
Design Studio	File, Import Channels	Displays the Import Channels dialog box.
Design Studio	File, Import Demand Points	Displays the Import Demand Points dialog box.

Importing Products

Access the Import Products dialog box.

To import products:

1. Navigate to the directory that contains the XML file with the product information that you want to import.
2. From the list of available files, select the XML file that you want to import, and then click Open.

Importing Locations

Access the Import Locations dialog box.

To import locations:

1. Navigate to the directory that contains the XML file with the location information that you want to import.
2. Select the XML file that you want to import from the list of available files.
3. Click Open.

Importing Channels

Access the Import Demand Points dialog box.

To import channels:

1. Navigate to the directory that contains the XML file with the channel information that you want to import.
2. Select the XML file that you want to import from the list of available files.
3. Click Open.

Importing and Exporting Demand Points

This section discusses how to:

- Import demand points.

- Export demand points.

Windows Used to Import and Export Demand Points

Form Name	Navigation	Usage
Design Studio	File, Import Demand Points	Opens the Import Demand Points dialog box.
Design Studio	File, Export Demand Points	Opens the Export Demand Points dialog box.

Importing Demand Points

Access the Import Demand Points dialog box.

To import demand points:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click OK.

Exporting Demand Points

Access the Export Demand Points dialog box.

To export demand points:

1. Complete these fields:
   - Save in
   - File name
   - Save as type
2. Click Save.

Identifying New Demand Points

Access the demand point that you want to identify as new.

To identify demand points as New:

1. Right-click the demand point and select Set New Flag.
2. Repeat step 1 for each demand point that you want to identify as new.

Clearing New Flags

Access the demand point whose New flag you want to clear.

To clear New flags:

1. Right-click the demand point and select Clear New Flag.
2. Repeat step 1 for each demand point whose New flag setting that you want to clear.

Note. Users with administrator privileges can also use DASH commands to clear the New flag on demand points.

See Also

Chapter 17, "Using the Demand Automation Shell," Business Model Management, page 120

Importing, Exporting, and Clearing Fixed Demand Points

This section discusses how to:

- Import fixed demand points.
- Export fixed demand points.
- Clear fixed demand points.

Windows Used to Import, Export, and Clear Demand Points

Form Name	Navigation	Usage
Design Studio	File, Import Fixed Demand Points	Opens the Import Demand Points dialog box.
Design Studio	File, Export Demand Points	Opens the Export Demand Points dialog box.

Importing Demand Points

Access the Import Demand Points dialog box.

To import demand points:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click OK.

Exporting Demand Points

Access the Export Demand Points dialog box.

To export demand points:

1. Complete these fields:
   - Save in
   - File name
   - Save as type
2. Click Save.

CHAPTER 7

Working with User-Defined Properties

This chapter provide an overview of user-defined properties and explains how to:

- Define property names.
- Change property names.
- Delete property names.

Understanding User-Defined Properties

Each product, location, and channel has a default set of property names associated with it. Any new demand model that you create uses the default property names. If the default property names do not suit the business needs, you can add, edit, or delete the property names for coordinates.

You normally define custom property names when you are setting up a demand model. You can also define property names when you create a new aggregation hierarchy or add a new product. After you have defined the property names, you can use them wherever the default properties are used, for example aggregation hierarchies, demand point sets, and so on.

In addition, you can also create classifications for products. For example, if an enterprise has a small group of products that consistently outsell the other products in a line, you can set up these classifications:

- Group A, which consists of the best-selling products.
- Group B, which consists of products that sell well but not as well as those in group A.
- Group C, which consists of products that have low sales volumes.

Property Name Definitions

You can define property names for products, locations, and channels. Depending on the properties that you want to work with, the system displays the appropriate properties dialog box. For example, to define property names for locations, you would access the Define Location Properties dialog box.

The procedures for defining property names are the same regardless of whether you are defining property names for products, locations, or channels. The property names that you define are not added to the demand model until you close the Define Properties dialog box.

Note. You cannot add property names that match existing property. In addition, you cannot use these property names because the Design Studio uses them internally; name, id, and top.

Defining Product Properties in Design Studio

This section discusses how to define product properties.

Windows Used to Work With Product Properties

Form Name	Navigation	Usage
Design Studio	Edit, Define Product Properties	Displays the Define Product Properties dialog box.
Design Studio	Edit, Define Location Properties	Displays the Define Location Properties dialog box.
Design Studio	Edit, Define Channel Properties	Displays the Define Channel Properties dialog box.

Defining Product Properties

Access the Define Product Properties, Define Location Properties, or Define Channel Properties dialog box.

To define property names:

1. Click Add.
2. Type the new property name in the field and select Enter.
3. Repeat steps 1 and 2 for each new property name that you want to define.
4. Click OK to save the changes.

Changing Property Names

Access the Define Product Properties, Define Location Properties, or Define Channel Properties dialog box.

You can change any of the property names used in a model. You typically do this when the current property names do not suit business needs.

Note. If you try to change a property name to a property name that already exists, the Design Studio displays an error message.

To change property names:

1. Select the property name that you want to change from the list.
2. Click Rename. The property name becomes editable.
3. Do one of the following:
   - To give the property a new name, type the new name and then press Enter.

- To modify the property name (for example, to change Customer to Customer Name), press the arrow key on the keyboard to move the cursor to the point in the name where you want to make the change. Type the addition to the name, and then press Enter.

Note. If you type a name that matches that of an existing property name, the system displays an error message and does not change the property name.

4. Repeat steps 1 to 3 for each property name that you want to change.
5. Click OK to save the changes.

Deleting Property Names

Access the Define Product Properties, Define Location Properties, or Define Channel Properties dialog box.

Note. You cannot delete property names that are being used in an aggregation hierarchy.

To delete property names:

1. Select the property name that you want to delete from the list and click Delete.
2. Click OK.

   If the property name is being used in an aggregation hierarchy, the system displays an error message, and does not delete the property name.

CHAPTER 8

Adding Products, Locations, and Channels

This chapter provides an overview of products, locations, and channels, and discusses how to:

- Add a product, location, or channel.
- Edit product, location, and channel properties.
- Delete products, locations, and channels.
- Add and maintain demand points.

Understanding Products, Locations, and Channels

You can add new products, locations, and channels to a data model. Products, locations, and channels define the framework for the business model. The combination of products, locations, and channels constitutes a demand point, which displays where potential demand for a product exists.

You can also identify products, locations, and channels as "new." You identify products, locations, and channels as new by setting their status using the Mark as New option. When you view products, locations, and channels, the flag appears in the Status column of the Demand Point Pane. Any products, locations, and channels that are marked as new are visible when you want to work with their histories in the Forecast Studio.

See Also

Selecting Product, Location, and Channel Levels from the Aggregation Hierarchy, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide*

Product, Location and Channel, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Forecast Studio Implementation Guide*

Filtering Demand Points, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Forecast Studio Implementation Guide*

Chapter 17, "Using the Demand Automation Shell," Understanding Business Model Management Commands, page 121

Adding a Product, Location, or Channel

This section discusses how to:

- Add a product.
- Add a location.

- Add a channel.

Windows Used to Add a Product, Location, or Channel

Form Name	Navigation	Usage
Design Studio	Insert, New Product	Displays the Insert New Product dialog box.
Design Studio	Insert, New Location	Displays the Insert New Location dialog box.
Design Studio	Insert, New Channel	Displays the Insert New Channel dialog box.
Design Studio	Insert, New Demand Point	Displays the Insert New Demand Point dialog box.
Design Studio	Edit, Update Properties of Coordinate	Displays the Update Product, Location, or Channel Properties dialog box.
Design Studio	Edit, Rename Properties of Coordinate	Displays the Rename Product, Location, or Channel Properties dialog box.

Adding Products

Access the New Product dialog box.

If you add a product, you must assign a demand point before you save the model so that you can view this information in the Consensus Conference Room. If a demand point is not assigned and the model is saved, then the product information cannot be viewed in the Consensus Conference Room.

To add a product:

1. In the Value column, select a field and type a value for the corresponding property name.
2. Click the Units of Measure tab.
3. Select the base unit of measure for the new product from available options in the Base Unit of Measure field.
4. If you want to assign one or more units of measure to the new product, select the units of measure from the Available list and then click Add.
5. Keep the Mark as New option selected if you want this product to be marked with the New setting in Design Studio.
6. Click OK.

Note. Do not use these characters as part of the property value of a product:

" (quote)

/ (slash)

+ (plus sign)

Using these characters can cause display errors in the Design Studio, the Forecast Studio, and the Consensus Conference Room.

See Also

Chapter 11, "Working with Units of Measure," Assigning Units of Measure to Demand Points, page 70

Adding Locations

Access the New Location dialog box.

If you add a location, you must assign a demand point before you save the model so that you can view this information in the Consensus Conference Room. If a demand point is not assigned and the model is saved, then the location information cannot be viewed in the Consensus Conference Room.

To add a location:

1. In the Value column, select a field and type a value for the corresponding property name.
2. Keep the Mark as New option selected if you want this location to be marked with the New setting in Design Studio.
3. Click OK.

Note. Do not use these characters as part of the property value of a location:

" (quote)

/ (slash)

+ (plus sign)

Using these characters can cause display errors in the Design Studio, the Forecast Studio, and the Consensus Conference Room.

Adding Channels

Access the New Channel dialog box.

If you add a channel, you must assign a demand point before you save the model so that you can view this information in the Consensus Conference Room. If a demand point is not assigned and the model is saved, then the channel information cannot be viewed in the Consensus Conference Room.

To add a channel:

1. In the Value column, select a field and type a value for the corresponding property name.
2. Keep the Mark as New option selected if you want this channel to be marked with the New setting in Design Studio.

3. Click OK.

Note. Do not use these characters as part of the property value of a channel:

" (quote)

/ (slash)

+ (plus sign)

Using these characters can cause display errors in the Design Studio, the Forecast Studio, and the Consensus Conference Room.

Editing Product, Location, and Channel Properties

Select the product, location, or channel that you want to edit, and then access the Properties dialog box.

To edit product, location, or channel properties:

1. In the Value column, select the field that you want to edit.
2. Type the change.
3. Click OK.

Deleting Products, Locations, and Channels

Access the lowest level that holds the product that you want to delete.

To delete a product, location, or channel

1. Select Delete from the Edit menu.
2. Click Yes.

Note. If any demand points are associated with the deleted product, location, or channel they are also deleted.

Adding and Maintaining Demand Points

This section provides an overview of demand points and discusses how to:

- Add a new demand point.
- Delete a demand point.
- Duplicate a product, location, or channel.
- Arrange demand point order in the Demand Point pane.

Understanding Demand Points

A demand point is a point in the business model where potential exists for a product to be sold from a location into a channel. This point is a combination of product, location, and channel; and is useful to help determine whether a company can sell products at a particular location and channel. For example, mountain bikes (product) sold in Boston (location) from a retailer in Mass Marketing (channel) represents a valid demand point.

Every demand point can have a forecast. Demand points at the lowest level of the business model (leaves) are combinations of individual products, locations, and channels. Demand points higher in the Business model are for sets of products, locations, and channels.

Users specify all of the leaf demand points. The demand points for levels that are higher in the business model are automatically created, based on the existence of points below them. Demand points (both aggregated and nonaggregated) are identified by their product, location, and channel paths.

You can also indicate that these demand points are new and require attention as they are marked new and appear bold. You can identify that a demand point is new by setting the status of the demand point to New. When you view a demand point, a visual cue appears in the Status column of the Demand Point Pane to indicate that the demand point is new. Normally, demands points are identified as new when they are imported into Design Studio. However, in Design Studio you can indicate that demand points are new to be able to work with their histories in the Forecast Studio.

The demand point statuses are:

- Aggregated

  Demand points whose data is summed up from lower demand points. Aggregated demand points enable you to see the total forecast or sales history for those demand points.

- Disaggregated

  Demand points that have their data prorated down from a higher abstraction level. With disaggregated demand points, you can compare forecast data, which was forecasted at a higher level, with another user's data that was forecasted at a lower level of abstraction.

- Save

  Demand points containing forecast data that is published to the database and cannot be changed by aggregation.

- New

  Demand points that have been added to the demand model, either manually or by being imported from an external system. Demand points that are identified as new require attention. In Design Studio, the system uses a visual cue to indicate that demand points are new.

If you add a product, location, or channel, you must assign a demand point before you save the model so that you can view this information in the Consensus Conference Room. If a demand point is not assigned and the model is saved, then the product, location, and channel information cannot be viewed in the Consensus Conference Room.

Demand points also have effective dates assigned to them. This identifies the period in which the demand point is available.

See Also

Chapter 6, "Introducing New Products," Understanding New Product Introduction, page 31

Filtering Demand Points, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Forecast Studio Implementation Guide*

Adding New Demand Points

Access the Insert New Demand Point dialog box.

To add a new demand point:

1. Select New Demand Point from the Insert menu.
2. Select a product, a location, and a channel.
3. Keep the Mark as New option checked if you want this demand point to be marked with the New flag in Design Studio.
4. Click Add, and then click Close.

Note. If a demand point already exists with the product, location, and channel that you specified, the add button becomes unavailable for input.

The demand point appears in the Demand Point pane under the selected product, location, or channel.

Deleting Demand Points

Select the demand point or demand points that you want to delete.

To delete a demand point, select Delete from the Edit menu.

Note. Deleting demand points from the model will also affect the same data that is open in the Forecast Studio and the Consensus Conference Room.

Duplicating Products, Locations, or Channels

You can associate a new product, location, or channel with the pattern of an existing product, location, or channel by duplicating the existing information and using it for the newly created product, location, or channel.

For example, a retailer in Mass Marketing wants to introduce a new style of mountain bike, model P809, into the market. The retailer in Mass Marketing is marketing the new mountain bike to the same audience as the existing model P808. The product, location, and channel information are exactly the same for both models. The retailer can use the existing model information for model P808, and simply duplicate and rename it for model P809.

To duplicate a product, location or channel:

1. Navigate to the lowest level of aggregation for one of the following:
   - Product
   - Location
   - Channel
2. Select Duplicate from the Edit menu.
3. Type a unique ID.
4. Type new property values for each field in the Value column.

   If you want to duplicate the existing values for the chosen product, location or channel, do not type new property values.
5. To duplicate the associated demand point data, click the Duplicate demand points from original option.

Note. If you select this option, you are associating a new demand point name with the pattern of an existing demand point.

6. To use the sales history from the original demand points with the new demand point, click the Duplicate history from original demand points option. If you do not select this option, you must associate a sales history with the demand point in the Forecast Studio.
7. If you do not want to mark the coordinate as new, click the Mark coordinate at New option.
8. Click OK.

See Also

Initializing Demand Points, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Forecast Studio Implementation Guide*

Chapter 6, "Introducing New Products," Identifying New Demand Points, page 34

Arranging Demand Point Order in the Demand Point Pane

You can change the order in which you view results in the Demand Point pane. As the default value, data in the view is sorted alphanumerically in descending order by the contents of the first column.

The data is immediately sorted in descending order by the specified column. Each time that you click a tab, the order of the list changes - for example, the alphabetical order changes from descending to ascending. Selecting a tab selects the field to sort. Selecting a tab a second time reverses the sort order.

To arrange demand point order in the Demand Point pane:

1. From the Demand Point list, select the name of the list item that you want to change positions.
2. Click on one of these headings for the column that you want to sort:
   - Product
   - Location
   - Channel
   - Unit of Measure

Updating and Renaming Products, Locations, or Channels

To help support product lifecycle management within your model, you can use the `rename_products`, `rename_channels`, and `rename_locations` DASH commands to rename existing products, locations, and channels. Renaming a property maintains your model structure and modifies the name for all products, locations, and channels containing that property value.

Alternately, you can use the `update_products`, `update_channels` and `update_locations` DASH commands to update existing products, locations and channels. Updating a coordinate modifies the model structure by creating a new hierarchy for the selected coordinate.

Updating Products, Locations, or Channels

To update a product, location or channel:

1. Navigate to the appropriate aggregation level for one of the following:
   - Product
   - Location
   - Channel
2. Select Update Properties of Coordinate from the Edit menu.
3. Type new property values for the appropriate field in the Value column.
4. Click OK.

See Also

Chapter 17, "Using the Demand Automation Shell," Updating Channels in the Demand Model, page 129

Chapter 17, "Using the Demand Automation Shell," Updating Locations in the Demand Model, page 130

Chapter 17, "Using the Demand Automation Shell," Updating Products in the Demand Model, page 130

Renaming Products, Locations, or Channels

To rename a product, location or channel:

1. Navigate to the appropriate aggregation level for one of the following:
   - Product
   - Location
   - Channel
2. Select Rename Properties of Coordinate from the Edit menu.
3. Type new property values for the appropriate field in the Value column.
4. Click OK.

   All products, locations, or channels containing the renamed property are changed.

See Also

Chapter 17, "Using the Demand Automation Shell," Renaming Channels in the Demand Model, page 136

Chapter 17, "Using the Demand Automation Shell," Renaming Locations in the Demand Model, page 137

Chapter 17, "Using the Demand Automation Shell," Renaming Products in the Demand Model, page 138

CHAPTER 9

Working With Demand Point Sets

This chapter provides an overview of demand point sets, and discusses how to:

- Define demand point sets by hierarchy.
- Define demand point sets by property.
- Create rule-based demand point sets.
- Convert to rule-based demand point sets.
- Edit demand point sets.
- Duplicate demand point sets.
- Delete demand point sets.

Understanding Demand Point Sets

This section provides an overview of demand point sets and discusses:

- Defining demand point sets by hierarchy.
- Defining demand point sets by property.
- Creating rule-based demand point sets.
- Converting to rule-based demand point sets.
- Editing demand point sets.
- Duplicating demand point sets.
- Deleting demand point sets.

Demand Point Sets

Access to information in a demand model is controlled by sets of demand points. You use the demand points to filter views and control access and report generation. You define demand point sets to filter information in a demand model so that members of the workgroup see only the information that is relevant to them, such as a particular product group in a particular channel. Setting access rights enables system administrators to enforce security down to the demand point level in the Design Studio, the Consensus Conference Room, and the Forecast Studio.

Demand Point Sets by Hierarchy

If you work with a large models, the ability to define demand points by hierarchy is especially useful. For example, instead of sorting through all products, locations, and channels to find the demand points, you can quickly access only the demand points that are relevant to you by filtering the demand points by hierarchy.

After demand points are filtered, you can add them to new or existing demand point sets. Defining demand points by hierarchy enables you to view demands points at a particular level in the model. The advantages of defining demand point sets by hierarchy extends beyond generating forecasts. For example, exception reports or forecast accuracy reports that other users create will contain only the forecasts that are relevant to them. This is because only the demand points that match the filtering criteria are included in the forecast data for the demand point set that the forecast provider uses.

Demand Point Sets by Property

Demand point sets can also be generated by querying demand points based on properties. After the list of demand points is generated, you must search through the list to locate the demand points that you want to use. You can do this by defining a filter.

Consider these points when filtering data for demand point sets:

- If you filter demand points using a property that is not available in an aggregation hierarchy, you will not see it in the property results.
- If a property is available, but no property value is assigned, you will not receive any results.

  However, if you filter based on the "does not contain" rule, the results will include demand points that contain the empty property value.
- Using the aggregation level is optional when you filter demand points by property.
- Aggregated points appear only at the level specified.

  For example, if you select the aggregation level *Product Group by Region*, the system displays demand points at this level and no others.
- Use an attribute once when filtering demand point sets.

  For example, only use the *Channel.customer* property in one rule.
- The system displays the filtered results, which indicate where demand points reside in the aggregation hierarchy.

For example, you can filter first on the *Group by Region* aggregation level. Next, you filter on property names that match the results found in the aggregation level filter. For example:

- *Location.region* is equal to *Toronto.*
- *Product.name* contains the word *Bike.*
- *Customer.name* is equal to *Independent.*

The system returns a list of filtered demand points at the aggregation level region, for all of the products containing the word Bike, sold by the customer Independent, and produced in the Toronto region.

Demand Point Set Generator Symbols

Several symbols appear on the Demand Point Set Generator. The various symbols are:

Product

 Location

 Channel

 Leaf Level

 Aggregation Level

Rule-Based Demand Point Sets

Non-rule-based demand points sets are static by nature. Rule-based demand points sets are dynamic in nature and are updated using a refresh. Rule-based demand point sets are generated by querying demand points based on specific property rules. Demand point sets that are based on property rules are generated and stored in the Demand Point Set Manager.

Rules can contain valid and invalid demand points. If a property no longer exists, any demand point set that contains this property becomes invalid.

When new demand points are added to the demand model it does not display until a manual update (using either a refresh operation or using DASH commands) is performed by the system administrator to ensure that any demand points that match the rules are added to the demand point set.

After rule based demand point sets are generated, you cannot edit or delete any demand point in the set. The summary displays both the rules and demand points that are only synchronized after a refresh operation takes place.

Conversion to Rule-Based Demand Point Sets

System administrators can change between rule-based and non-rule-based demand point sets. Changing the type of the demand point affects the contents of the set as follows:

- When changing from rule-based to demand point sets, all rules are deleted and all demand points are kept.
- When changing from demand point sets to rule-based demand point sets, all demand points are kept.

Duplication of Demand Point Sets

Manually creating demand point sets can be a lengthy, labor-intensive process. Because demand point sets often have similar attributes, such as common aggregation levels, you can save time and effort when creating demand point sets by duplicating existing parent demand point sets and changing only the elements that are different, such as the specific products, locations, or channels that make up the set. After you have duplicated a demand point set, you can edit or rename the new set.

The system adds "Copy of" before the name of the duplicate demand point set. For example, if the original demand point set is named Northwest Sales, the duplicate is named Copy of Northwest Sales.

Defining Demand Point Sets by Hierarchy

This section discusses how to define demand point sets by hierarchy.

Windows Used to Define Demand Points by Hierarchy

Form Name	Navigation	Usage
Demand Point Set Manager	Edit, Demand Point Sets	Displays the Demand Point Set Manager.
Demand Point Set Wizard	Edit, Demand Point Sets, Add	Displays the Demand Point Set Wizard.

Defining Demand Point Sets by Hierarchy

Access the Demand Point Set Wizard.

Prerequisite

Open the demand model to which you want to assign demand point sets.

To define demand point sets by hierarchy:

1. Complete these fields:
   - Name
   - Description
2. Click Next.
3. Complete these fields:
   - Aggregation Level
   - Product
   - Location
   - Channel
4. Select one of these options:
   - This Demand Point Only

     This option enables you to access the demand point specified by the Product, Location, and Channel combination.
   - Include Siblings

     This option enables you to access the specified demand point and any demand points that are on the same aggregation level.
   - Include Children

     This option enables you to access the specified demand point and any demand points that are under it. You can specify the number of levels below the current demand point to use with the Level(s) list.
5. Click Filter.
6. Select the demand points that you want to use, and then click one of these buttons:
   - Add selected
   - Add all
7. Repeat steps 3 to 6 for any other demand points that you want to allow the user to access.
8. Click Next to view a summary of the demand points you chose.

9. Click Finish.

See Also

Chapter 17, "Using the Demand Automation Shell," Demand Point Sets Management, page 217

Defining Demand Point Sets by Property

Access the Demand Point Set Wizard.

To define demand point sets by property:

1. Complete the Name field.
2. Optionally, type a description of the demand point set in the Description field.
3. Click Next.
4. Click the Demand Points by Property tab and complete these fields:
   - Property Name: Select *Aggregation Level.* You base the query on the aggregation level. Properties with a green check mark are valid.
   - Rule: Select one of these values from the shortcut menu:

     - *is equal to*: Only demand points containing the criteria in the Property Value field will be included in the list.

     - *not equal to*: All demand points, other than the ones containing the criteria in the Property Value field, will be included in the list.

     - *Contains*: Demand points will be included in the list if they contain the criteria in the Property Value field.

     - *does not contain*: All demand points, except those containing the criteria in the Property Value field, will be included in the list.
5. To complete the Property Value field, either type the filtering criteria in the field, or click the button that is next to the field and choose a value from the dialog box that appears.
6. Click OK.

Note. Click the plus button next to the Property Value field to add an additional property value to a rule. Adding more property values enables you to specify multiple values for a particular property name, and filter on one value or the other.

---

7. Click Add Rule and complete the Property Name field with the property that you want to filter on.
8. Repeat steps 4 to 7 for each new filter you want to build.
9. Click Filter.
10. Do one of the following:
    - Click Add all.
    - Select the demand points that you want to use, and then click Add selected.
11. Click Next.

12. Review the demand point set and click Finish.

Creating Rule-Based Demand Point Sets

Access the Demand Point Set Wizard.

To create a new rule-based demand point set:

1. Click Add.
2. Complete these fields:
   - Name
   - Description
3. Click the option next to Rule-Based, and then click Next.
4. Complete these fields:
   - Property: Select a property name.

     Properties with a green check mark are valid.
   - Function: Select one of these values from the shortcut menu:

     - *is equal to*: Only demand points containing the criteria in the Property Value field will be included in the list.

     - *not equal to*: All demand points, other than the ones containing the criteria in the Property Value field, will be included in the list.

     - *Contains*: Demand points will be included in the list if they contain the criteria in the Property Value field.

     - *does not contain*: All demand points, except those containing the criteria in the Property Value field, will be included in the list.
5. To complete the Value field, type the filtering criteria in the field or click the button next to the field and choose a value from the dialog box that appears.
6. Click OK.

Note. Click the plus button next to the Property Value field to add an additional property value to a rule. Adding more property values enables you to specify multiple values for a particular property name. You can then filter based on one of the values.

---

7. Click Add Rule for each new rule that you want to add and repeat steps 5 to 7.
8. Click Refresh.
9. Click Next.
10. Review the new demand point set and click Finish.

Converting to Rule-Based Demand Point Sets

Access the Demand Point Set Manager.

To convert demand point sets into a new rule-based demand point set:

1. Select the demand point set that you want to change and click Edit.
2. Complete these fields:
   - Name
   - Description
3. Click the option next to Rule-Based.

   You can change the type of demand point set using this option, for example from rule-based demand point set to demand point set. Any invalid properties appear with a red check mark. The system disables the Refresh and Next buttons until all invalid rules are cleared.
4. Click Next.
5. Complete these fields:
   - Property: Select a property name.

     Properties with a green check mark are valid.
   - Function: Select one of these values from the shortcut menu:

     - *is equal to*: Only demand points containing the criteria in the Property Value field will be included in the list.

     - *not equal to*: All demand points, other than the ones containing the criteria in the Property Value field, will be included in the list.

     - *Contains*; Demand points will be included in the list if they contain the criteria in the Property Value field.

     - *does not contain*: All demand points, except those containing the criteria in the Property Value field, will be included in the list.
6. To complete the Property Value field, either type the filtering criteria in the field, or click the button next to the field and choose a value from the dialog box that appears.
7. Click OK.

   Note. Click the plus button next to the Property Value field to add another property value to a rule. Adding more property values enables you to specify multiple values for a particular property name. You can then filter on the individual values.
8. Click Add Rule and repeat steps 4 to 7 for each new rule that you want to add.
9. Click Refresh.
10. Click Next.
11. Review the new demand point set and click Finish.

Editing Demand Point Sets

You can edit demand points sets. For example, you can:

- Add new demand points to a demand point set when new products, locations, and channels are introduced into a demand model.
- Remove demand points from a demand point set when older products are phased out, or when a location stops carrying a particular product.
- Rename demand point sets when you have duplicated a demand point set or want to give an existing demand point set a more descriptive name.
- Refresh rule-based demand point sets.

Adding New Demand Point Sets

Access the Demand Point Set Manager.

To add new demand points to a demand point set by hierarchy:

1. Select the demand point set to which you want to add new demand points and then click Edit.
2. Click Next.
3. Select the Aggregation Level field with the level of aggregation for the demand point.
4. Complete these fields:
   - Product
   - Location
   - Channel
5. Select one of these options:
   - *This Demand Point Only*: Enables the user to access the demand point specified by the combination of product, location, and channel.
   - *Include Siblings*: Enables the user to access the specified demand point and any demand points that are at the same aggregation level.
   - *Include Children*: Enables the user to access the specified demand point and any demand points that are under it.

     You can specify the number of levels below the current demand point to use with the Level(s) list.
6. Click Filter.
7. Click one of these buttons:
   - Add selected
   - Add all
8. Click Next.
9. Click Finish.

Removing Demand Points from Demand Point Sets

Access the Demand Point Set Manager.

To remove demand points from a demand point set:

1. Select the demand point set that you want to remove, and then click Edit.
2. Click Next.
3. Select the demand point that you want to remove.
4. Click Remove Selected, and then click Next.
5. Click Finish to save the changes.

Renaming Demand Point Sets

Access the Demand Point Set Manager.

To rename demand point sets:

1. Select the demand point set or rule based demand point set that you want to rename, and then click Edit.
2. Complete these fields:
   - Name
   - Description
3. Click Next.
4. Click Next again.
5. Click Finish, and then click Close.

Refreshing Demand Point Sets

Access the Demand Point Set Manager.

To refresh a rule-based demand point set:

1. Select a rule-based demand point set.

   The demand point set must indicate *Yes* under the Valid column, indicating that all rules are valid.
2. Click Refresh.
3. Click Close.

See Also

Duplicating Demand Point Sets

Access the Demand Point Set Manager.

To duplicate a demand point set:

1. Select the demand point set or rule based demand point set that you want to duplicate.
2. Click Duplicate.

Deleting Demand Point Sets

Access the Demand Point Set Manager.

Important! You cannot recover deleted demand point sets.

To delete demand point sets:

1. Select the demand point set or rule based demand point set that you want to delete and click Delete.
2. Click Yes.
3. Repeat steps 1 and 2 for each demand point set that you want to delete.

CHAPTER 10

Working with Effective Dates

This chapter provides an overview of effective date, and discusses how to:

- Import effective dates.
- Export effective dates.
- Set effective dates for demand points.
- Reset effective dates for demand points.
- Remove effective start and end dates.

Understanding Effective Dates

Effective dates indicate when demand points are available or unavailable in the forecasting process. Effective dates are set and changed in the demand model using the Design Studio. You can set the effective start date and end date for a single demand point or for multiple demand points.

Setting effective dates for demand points benefits the forecasting process in these instances:

- Products are not available at a location during a specific date period.
- Channels do not carry a product during a specific date period.
- Product lines are discontinued or reintroduced on a specific date.

Knowing the effective start and end dates helps to assure demand planners that forecasting data is valid. To avoid generating demand point exceptions in the forecasting process, demand planners should apply effective dates to demand points.

Guidelines for Setting Effective Dates

Use these guidelines for setting effective dates:

- Dates apply to demand points at the lowest levels of aggregation (leaf level).
- Demand points can become effective during and extending past the forecast horizon date.
- Demand point values that fall outside of the effective date range appear as highlighted.
- If an effective date starts beyond the beginning of a time bucket, the entire time bucket and data are included in the forecast.

  For example, if time buckets are measured in monthly increments and the effective date starts on the 5th of the month, the entire monthly time bucket of forecast data is included.
- Setting a specific effective date for demand points can affect exception report generation.

- Missing forecast data is included in the effective dates.

Set Effective Dates

You can specify whether the demand points are available or unavailable to the forecasting process at certain dates within the process. Setting these dates in the demand model ensures that this information is available in either the Consensus Conference Room or Forecast Studio.

You can set the effective dates for demand points by choosing from these options:

- Start date only.
- End date only.
- Both the start and end date.

Reset Effective Dates

You can reset the start and the end effective dates for demand points by selecting:

- A single demand point.
- Multiple demand points.

Multiple demand points that already have different effective dates assigned appear on the Effective Dates dialog box, with the start date option chosen but unavailable. The default start date is the first date of the forecast horizon. Resetting effective dates for demand model data can affect data in other the Consensus Conference Room and Forecast Studio.

Remove Effective Dates

You can remove effective start and end dates from demand points in the demand model that is open in the Design Studio. Removing effective dates for demand model data can affect data in the Consensus Conference Room and Forecast Studio.

Importing Effective Dates in Design Studio

This section discusses how to import effective dates.

Windows Used to Import Effective Dates

Form Name	Navigation	Usage
Design Studio	File, Import Effective Dates	Displays the Import Effective Dates dialog box.
Design Studio	File, Export Effective Dates	Displays the Export Effective Dates dialog box.
Design Studio	Edit, Effective Dates	Displays the Effective Date dialog box.
Design Studio	File, Reset All Effective Dates	Displays the Reset All Effective Dates dialog box.

Importing Effective Dates

Access the Import Effective Dates dialog box.

You can import effective dates for demand points from an XML file that resides on a local drive or on a network drive. Importing these dates into the model from an external source overwrites existing effective dates. Importing a file that contains effective dates ensures that this information is available in either the Consensus Conference Room or the Forecast Studio.

To import effective dates:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click Open.

Exporting Effective Dates

Access the Import Effective Dates dialog box.

You can export effective dates from the Design Studio to an XML file that resides on a local drive or on a network drive. You can import the resulting XML file into another application, such as EnterpriseOne ERP software, or use the data with other demand models.

To export effective dates:

1. Complete these fields:
   - Look in
   - File name
   - File type
2. Click Save.

Setting Effective Dates for Demand Points

Select the demand points for which you want to set the effective start and end dates, and access the Effective Date dialog box.

To set the effective start and end dates for demand points:

1. Click the option next to the Becomes Effective on field.
2. Complete the Becomes Effective on field.
3. Click the option next to the Expires On field to set the effective end date.
4. Complete the Expires On field.
5. Click OK.

Note. Previously set effective dates for a single demand point appear in the effective dates list. If you select multiple demand points with different effective dates, the Effective Dates dialog box is temporarily unavailable until you click one of the options.

Resetting the Effective Dates for Demand Points

Select the demand points for which you want to reset the effective start and end dates, and then access the Reset All Effective Dates dialog box.

To reset the effective date for demand points, click Yes to reset the effective dates for all of the demand points that appear in the Demand Point pane.

The system replaces effective dates with a dash (-).

Removing Effective Start and End Dates

Select the demand points for which you want to remove the effective start and end dates, and then access the Effective Dates dialog box. When you remove effective dates, the Design Studio replaces the effective dates with dashes.

To remove effective start and end dates from demand points:

1. Clear the option next to the Becomes effective on field.
2. Clear the option next to the Expires on field.
3. Click OK.

See Also

Chapter 17, "Using the Demand Automation Shell," Effective Date Management, page 162

CHAPTER 11

Working with Units of Measure

This chapter provides an overview of units of measure and discusses how to:

- Add units of measure.
- Rename units of measure.
- Delete units of measure.
- Change the base unit of measure
- Work with conversion rates.
- Import and export conversion rates.
- Assign and remove units of measure.

Understanding Units of Measure

Different company areas track products by using different measuring schema. For example, the marketing department might compile forecast data using dollars, while the sales department might compile forecast data using pallets as a unit of measure. To compare both sales and marketing forecast data, one common unit of measure must be chosen.

You can create a custom unit of measure conversion table that maps each unit of measure that is used in the business model to a specific value. The table converts between units that use the same system of measure - for example, grams and kilograms - and those that are disparate, such as dollars and pallets. Each unit of measure is defined with a name and a value rate.

For example, one American dollar is the base conversion rate for all other units. One pallet is equal to $100.00 American dollars (USD). If seven pallets are converted to American dollars, the conversion appears as follows:

(1 pallet = $100.00 American dollars (7 pallets x $100.00 = $700.00) American dollars)

If more than one kind of unit of measure is used for the same system - for example, dollars - you can create different currencies and their value rates. One Canadian dollar (CAD) has a different value than that of one USD. This difference is reflected in the conversion table.

Units of measure are applied to each demand point at the lowest leaf level in the business model. Even though some demand points share the same unit of measure name, the value rate can be different. For example, a car that is sold at one demand point might have the value rate of 25,000 USD. A car sold at a different demand point level in the business model might have the value rate of 30,000 (CAD). Even though both cars are sold in dollar values, the value rate between the currencies is different.

Note. You can also change the base unit of measure for a demand point if you need to change the way in which a conversion rate is calculated for that demand point.

See Also

Chapter 11, "Working with Units of Measure," Changing the Base Unit of Measure, page 67

Dynamic Units of Measure

Product prices can vary over time due to such factors as promotions, fluctuations in exchange rates, and so on. To keep up with these changes, you can dynamically set prices and conversion rates for products at specific demand points within the Design Studio. Dynamically setting units of measure like conversion rates and prices enables you to:

- Trace the effects of price changes on demand for a product.
- Set one or more conversion rates for products at specified periods in the horizon of a demand model.

When working with dynamic unit of measure data, you would typically generate a forecast for a particular demand point, for example in dollars. Then, to see the forecast in other units of measure, you recreate the original forecast, but this time set the unit of measure to pallets. You do this by applying the conversion rate on a bucket per bucket basis, while respecting the effective dates on which the conversion rate is active.

Note. Dynamic unit of measure information, including price, is not stored as a time series.

Base Unit of Measure Changes

The base unit of measure for a product is the basis for converting between units that use either the same units of measure, or disparate units of measure. All conversion rates are relative to the base unit of measure for a product. When you change the base unit of measure of a product, it effects the way in which the conversion rate used by that product is calculated.

Note. Any change to the base unit of measure overrides the conversion rates for the product, and resets the conversion rates to 1. Any dynamic product pricing information that is associated with the product is lost when you change the base unit of measure.

Conversion Rates

You can change the conversion rates in a demand model in two ways:

- You can manually add, edit, or delete the conversion rates for individual demand points from within the Design Studio.
- You can import an XML file that contains the new conversion rates for the specified demand points.

Add Conversion Rate

You can add conversion rates to demand points. The basis for new conversion rates is the base unit of measure for the demand point. When adding a conversion rate, you must specify:

- A unit of measure for conversion.
- The date when the new rate takes effect.

  The effective date for a new conversion rate can be anywhere in the horizon for the demand model. The default effective date is the beginning of the horizon
- A value for the conversion.

  This value determines the amount by which the base unit of measure is multiplied to equal the unit of measure for the new conversion rate. The default effective value is 1.

When you add a new unit of measure to the demand model, that unit of measure has a conversion rate of 1.0 at all demand points. The first unit of measure that you add to a demand model becomes the base unit of measure for all of the demand points in the model.

Note. You cannot add a conversion rate with the same values as a conversion rate that already exists in the demand model.

Edit Conversion Rates

You can edit the properties of existing conversion rates to change their effective dates and effective values if you discover that the conversion rates that you set are affecting forecasts. You cannot, however, change the effective date of the default conversion rate, which is always the beginning of the horizon.

Delete Conversion Rates

You can delete conversion rates that are no longer effective, or that you created and have not used or do not plan to use. After you delete conversion rates, you cannot recover them. In addition, you cannot delete the default conversion rate for a demand point.

Conversion Rate Imports and Exports

Instead of manually entering conversion rate information, you can import conversion rates from and export to an XML file. To exchange conversion rates between demand models, each demand model must contain the same set of demand points. If the demand models do not contain the same set of demand points, the import process fails.

Import Conversion Rates

Instead of manually entering a set of conversion rates, you can import the rates from an XML file. You normally create the XML file by exporting the conversion rates from an existing demand model, or by using enterprise data exported from another system.

The XML file that contains the conversion rates information must conform to a specific structure. The base unit of measure must be specified first, and all of the entries in the file must be in chronological order. If these criteria are not met, then the import process will fail.

The import process also fails if the demand model contains fewer than two units of measure or if a specific demand point listed in the conversion rates file is not found in the demand model that is currently loaded in the Design Studio.

Export Conversion Rates

You can export conversion rates from a demand model to an XML file. Exporting conversion rates enables you to:

- Use the conversion rates with another demand model.
- Back up the conversion rates information in case you want to later revert to the old rates.

The export process fails if the demand model from which you are exporting contains fewer than two units of measure.

Unit of Measure Assignment

A single unit of measure may not be applicable to all of the items across a product line. For example, if a company sells both bottled water and processed foods, the unit of measure for the water might be bottles and the unit of measure for the food might be boxes. In such cases, using the same unit of measure is not practical.

In addition, all of the units of measure that are common to the child demand points in an aggregated demand point may not be applicable to all points. For example, suppose that the bottled water and processed foods are stacked in pallets in a warehouse. The pallets of bottled water are clearly different from the pallets of food. In this case, the aggregated demand points for one or both of these products should not be measured in pallets.

Unit of measure assignment enables you to control whether particular units of measure are applicable at the demand-point level. This prevents irrelevant units of measure from being included in forecast calculations, and it stops inappropriate units of measure from being displayed in the application.

The units of measure that are available in the Design Studio are defined when the application is implemented in an enterprise. If the units of measure that you want to use were not set up during implementation, you must define the units of measure.

The assignment of units of measure to demand points is a bottom-up process. You assign the units of measure first to products at the demand-point level. After the units of measure are assigned to individual demand points, they become available at the product-group level. Ensure that the unit of measure that you assign to a particular demand point is applicable to that demand point. For example, you would not assign kilogram as a unit of measure for a bicycle.

Note. You can also assign the units of measure to other leaf-level demand points or to the parents of the leaf-level demand points.

Although assigning units of measure to demand points is a bottom-up process, removing units of measure from demand points is often a top-down process. This is necessary because, in many cases, you assign units of measure to both leaf demand points and parent demand points. You must delete the demand points starting from the top of the tree in order to keep the demand point tree consistent.

Note. If the Design Studio displays the following error message, you must remove the unit of measure from each of the parent demand points to which it is assigned before removing it from the leaf demand point.

Adding Units of Measure

This section discusses how to add a unit of measure.

Window Used to Add a Unit of Measure

Form Name	Navigation	Usage
Design Studio	Edit, Units of Measure	Displays the Units of Measure dialog box.

Adding Units of Measure

Access the Units of Measure dialog box.

To add a unit of measure:

1. Click Add.

2. Type a name for the new unit of measure, and then click Close.

Note. These characters are not supported:

" (quote)

/ (slash)

+ (plus sign)

Renaming Units of Measure

Access the Units of Measure dialog box.

To rename a unit of measure:

1. Select a unit of measure and click Rename.
2. Type a new name for the unit of measure, and then select Enter.
3. Click Close twice.

Deleting Units of Measure

Access the Units of Measure dialog box.

To delete a unit of measure:

1. Select a unit of measure and click Delete.
2. Click Yes.
3. Click Close.

Changing the Base Unit of Measure

This section discusses how to change the base unit of measure.

Windows Used to Change the Base Unit of Measure

Form Name	Navigation	Usage
Design Studio	Edit, Conversion Rates	Displays the Conversion Rates dialog box.
Design Studio	File, Import Conversion Rates	Displays the Import Conversion Rates dialog box.
Design Studio	File, Export Conversion Rates	Displays the Export Conversion Rates dialog box.

Changing Base Units of Measure

Access the product whose base unit of measure you want to change.

To change the base unit of measure:

1. Right-click the product, and then select Base Unit Of Measure.
2. Select a new base unit of measure from the available options.
3. Click OK.

Working with Conversion Rates

This section discusses how to:

- Add conversion rates.
- Edit conversion rates.
- Delete conversion rates.

Adding Conversion Rates

Access the Conversion Rates dialog box.

To add conversion rates:

1. If necessary, select a unit of measure for the conversion from the available options in the Unit of Measure field.
2. Click New.
3. Complete the Effective Date field.

   Note. You cannot specify a date that is outside of the horizon for the demand model.

4. Specify a new conversion rate in the Effective Value field.

   The value that you enter determines the amount by which the base unit of measure is multiplied to equal the unit of measure for the new conversion rate. For example, if the base unit of measure is Each and you want to convert to Pallets, enter the number of eaches in a pallet.
5. Click OK to save the new conversion rate.
6. Click Apply to apply the new conversion rate.
7. Click OK.

Editing Conversion Rates

Access the Conversion Rates dialog box.

To edit conversion rates:

1. Select the conversion rate that you want to edit and click Edit.
2. Complete these fields:

- Effective Date
- Effective Value

  This value determines the amount by which the base unit of measure is multiplied to equal the unit of measure for the new conversion rate. For example, if the base unit of measure is Each and you want to specify rates for Pallets, enter the number of pallets that equals one Each.

3. Click OK.
4. Click Apply.
5. Click OK.

Deleting Conversion Rates

Access the Conversion Rates dialog box.

To delete conversion rates:

1. Select the conversion rate that you want to delete.
2. Click Delete.
3. Click Yes, and then click Apply.
4. Click OK.

Importing and Exporting Conversion Rates

This section discusses how to:

- Import conversion rates.
- Export conversion rates.

Importing Conversion Rates

Access the Import Conversion Rates dialog box.

To import conversion rates:

1. Navigate to the directory containing the conversion rates XML file and select the file.
2. Click Open.

Exporting Conversion Rates

Access the Export Conversion Rates dialog box.

To export conversion rates:

1. Navigate to the directory where you want to save the conversion rates file.
2. Type a name for the conversion rates file in the File name field.
3. Click OK.

Assigning and Removing Units of Measure

Assigning units of measure to individual demand points can be a tedious process if you assign the same unit of measure to multiple demand points. You can push a unit of measure down to products in the product tree of the Design Studio. Although you can only assign one unit of measure at a time in this way, that unit of measure is applied to all of the products in the sub-tree.

There is no way to globally undo the unit of measure assignment. You must manually remove units of measure.

Windows Used to Assign and Remove Units of Measure

Form Name	Navigation	Usage
Assign Units of Measure	Edit, Assign Units of Measure	Add existing units of measure to individual products.
Assign Units of Measure to Sub Tree	Edit, Assign Units of Measure to Sub Tree	Add existing units of measure to all of the products in a demand point.

Assigning Units of Measure to Demand Points

Select a demand point, and then access the Assign Units of Measure window.

To assign units of measure to demand points:

1. Select one or more units of measure from the Available column and then click Add.
2. Click OK.
3. Repeat steps 1 and 2 for each product and product group to which you want to assign a unit of measure.

Assigning Units of Measure to a Sub-Tree

Select a demand point, and then access the Assign Units of Measure to Sub Tree window.

To assign units of measure to demand points:

1. Select one or more units of measure from the unit of measure list.
2. Click Assign, and then click Close.

Removing Units of Measure from Demand Points

Select a demand point, and then access the Assign Units of Measure window.

To remove units of measure from demand points:

1. Select the unit of measure that you want to remove.
2. Click Remove.
3. Click OK.

CHAPTER 12

Forecast and History Horizons

This chapter provides an overview of forecast and history horizons and discusses how to set forecast horizon values.

Understanding Forecast and History Horizons

Once you have set the horizons for the business model, all demand - including forecasts and historical demand - is stored internally in the calculated time buckets. If you import forecast or sales history data by using the same horizon as the business model, the data is stored internally exactly as it is imported. However, you can import forecasts with a horizon that is totally different from the business model. In such a case, the forecast values are re-bucketized and fit into the business model horizon structure.

These guidelines explain how the system processes horizon data:

- If any of the forecast data falls outside the horizon's start and end dates, the data outside the horizon is discarded.
- If the data is at a more granular level than data in the time buckets of the business, the values are aggregated into the buckets of the business.

  For example, when daily data is imported into a business model with weekly time buckets the daily demand is aggregated into weekly demand.
- If the data is at a larger granular level than data in the business model, it is broken down by using a fair-share algorithm.

  For example, when weekly data is imported into a business model with daily time buckets, each day gets 1/7 of the weekly demand.

If the imported data crosses time bucket boundaries, it is apportioned to the correct time buckets within the business model.

Setting Forecast Horizon Values

This section discusses how to set forecast horizon values.

Window Used to Set Forecast Horizon Values

Form Name	Navigation	Usage
Design Studio	Edit, Horizon	Displays the Horizon dialog box.

Setting Forecast Horizon Values

Access the Horizon dialog box.

When you set forecast horizon values, the end date is automatically set based on the start date and number of buckets. In addition, the historical start date and end date are automatically calculated using the forecast start and end date.

To set forecast horizon values:

1. Select the type of time period in which you want the data measured from the Time Bucket Type field. Options are:
   - *Daily*
   - *Weekly*
   - *Monthly*
   - *Yearly*
   - *Four Weekly*
   - *Weekly 445*
   - *Weekly 544*
2. Select the start date of the forecast data.

   To select the date:

   a. Click the default year, and use the up and down arrows to adjust the setting.

   b. Click the default month, and use the up and down arrows to adjust the setting.

   c. Click the default day, and use the up and down arrows to adjust the setting.
3. Select the number of buckets that the forecast can span from the available options in the Number of Time Buckets field.
4. Select the number of buckets that the historical data can span from the available options in the Number of Time Buckets field.
5. Click OK to set the horizon values.

CHAPTER 13

Working With Aggregation Hierarchies

This chapter discusses how to:

- Add an aggregation hierarchy.
- Rename aggregation levels.
- Edit an aggregation hierarchy.
- Make aggregation hierarchy the fixing hierarchy.

Understanding Aggregation Hierarchies

This section provides an overview of aggregation hierarchies, multiple aggregation hierarchies, and the Aggregation Wizard.

Aggregation Hierarchies

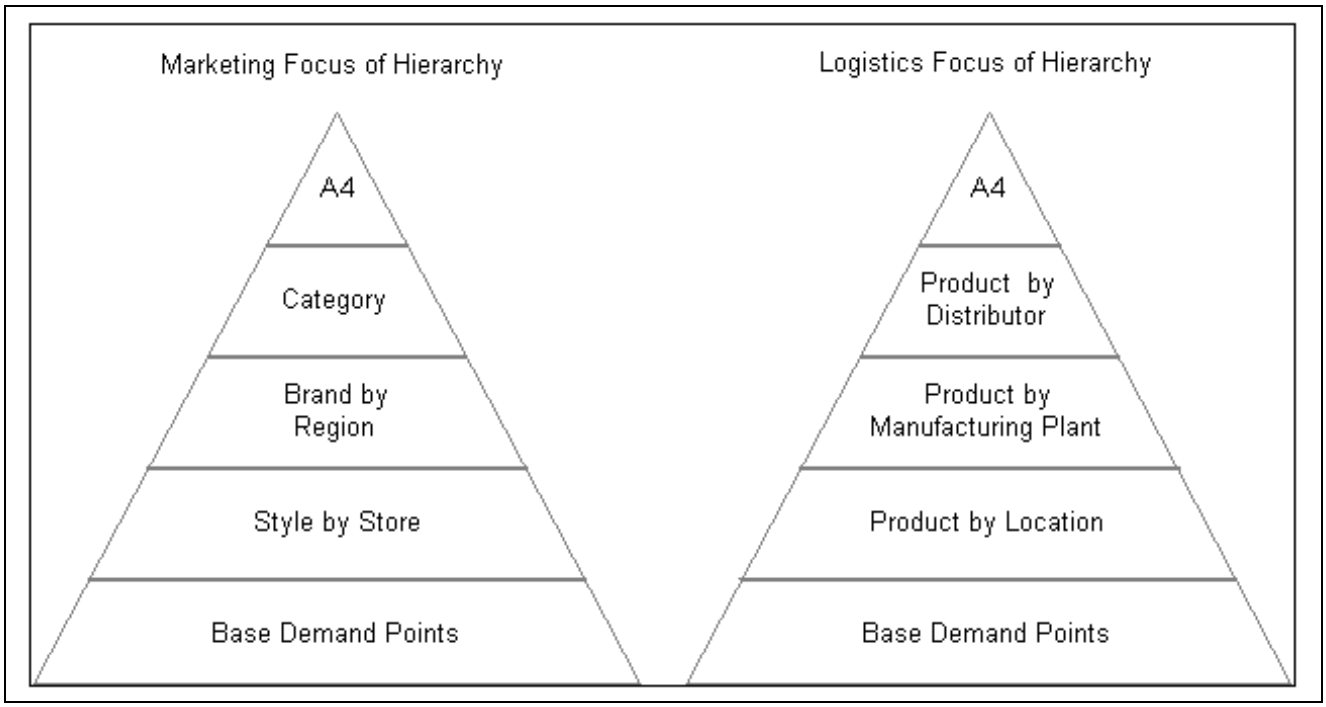
An aggregation hierarchy provides a view of the business model according to how you arrange the details. Aggregating a business model enables you to review the total forecast or sales history for a set of demand points. Aggregation automatically calculates a value for a demand point by summarizing the associated values from the demand points in the next lower-level aggregation. You can specify that one aggregation hierarchy in your model is the "fixing hierarchy", meaning that you can fix demand points within that aggregation hierarchy. Once set, the fixing hierarchy can be applied to a different aggregation hierarchy only if no fixed demand points are present in the current fixing hierarchy.

Users can view aggregation hierarchies for which they have permissions. The demand point data at the lowest level are the same for each aggregation hierarchy. When you switch between aggregation hierarchies, the aggregation levels and demand points that appear in the aggregation display different perspectives in the data.

Multiple Aggregation Hierarchies

Creating multiple aggregation hierarchies enables you to create several different views of the business model. When forecast data is used with the business model, it then becomes a demand model. For example, within a demand model, forecast data for which the marketing department is responsible is configured differently than that for which the logistics department is responsible. The marketing department is typically interested in data at higher levels of abstraction, whereas the logistics department wants to configure forecast data at a more granular level. Users can configure their aggregation hierarchy to suit their individual forecasting needs by customizing their view of the forecast data from the base demand model.

This is illustrated in this diagram:



Multiple aggregation hierarchies

See Also

Chapter 6, "Introducing New Products," Understanding New Product Introduction, page 31

The Aggregation Wizard

The Aggregation Wizard is a tool that enables you to specify the aggregation hierarchy for the business model. You can add, edit, rename, and delete aggregation hierarchies.

After you define the aggregation hierarchy name, you define the product, location, and channel levels. You define an aggregation level in the aggregation hierarchy by choosing one each of the previously defined product levels, location levels, and channel levels.

For each aggregation level, you can review and manipulate the aggregated demand for the cross-product of all of the values on the chosen product, location, and channel properties. The wizard provides you with lists from which you can select properties when you create the aggregation hierarchy. You need to change at least one property to add another level to the aggregation hierarchy. You can use the same properties in one or two of the lists when creating the aggregation hierarchy.

To create an aggregation hierarchy, you create a nested set of the aggregation levels. Using this type of hierarchy, you can refer to entire sets of products, locations, or channels in the same way that you would refer to a single product, location, or channel. For example, assume that the channel level is *top*, the location level is *region*, and the product level is *product family*. In this example, you would see the aggregated demand for each product family, at each region, summarized by all channels.

You can add one or several aggregation hierarchies for a single business model. The Aggregation Wizard guides you through the necessary steps required to define the aggregation hierarchy. You first define the aggregation hierarchy name, after which you define the attributes, such as levels for product, location, and channel, for the aggregation.

Adding an Aggregation Hierarchy

This section discusses how to add an aggregation hierarchy.

Windows Used to Add an Aggregation Hierarchy

Form Name	Navigation	Usage
Aggregation Hierarchies	Edit, Aggregation Hierarchies	Displays the Hierarchies list.
Aggregation Hierarchies Wizard	Edit, Aggregation Hierarchies, Add	Displays the Aggregation Hierarchies wizard.
Aggregation Names	Edit, Aggregation Names	Displays the Aggregation Names window.

Adding Aggregation Hierarchies

Access the Aggregation wizard.

To add an aggregation hierarchy:

1. Type a name for the new aggregation hierarchy and then click Next.
2. Select the product properties that you want to add to the Selected list from the Available list.
3. Click Add, and then click Next.
4. Select the location properties that you want to add from the Available list.
5. Click Add and then click Next.
6. Select the channel property that you want to add from the Available list.
7. Click Add and then click Next.
8. Arrange the classification property names into an aggregation hierarchy by choosing at least one item from one of these fields:
   - Product
   - Location
   - Channel

   Note. The default starting position is Top/Top/Top, which represents all products and locations, and the channel in the aggregation hierarchy. Classification levels that are not available options are unavailable for input.
9. Click Next.
10. Click Finish to build the aggregation hierarchy.

See Also

Chapter 13, "Working With Aggregation Hierarchies," Adding an Aggregation Hierarchy, page 75

Fixing Aggregation Hierarchy

Access the Aggregation Hierarchies window.

To make an aggregation hierarchy the fixing hierarchy:

1. Select an aggregation hierarchy that you want to make the fixing hierarchy.
2. Click Fixing Hierarchy.

   A check mark appears beside the selected aggregation level.

Renaming Aggregation Levels

Access the Aggregation Names window.

To rename aggregation levels:

1. Select an aggregation level that you want to rename.
2. Type a new name for the aggregation level and click OK.

Editing Aggregation Hierarchies

Access the Hierarchies list.

Note. Whenever you add or remove properties from the list, you must rebuild the aggregation levels.

To edit an aggregation hierarchy:

1. Select a hierarchy name from the list and click Edit.
2. Revise any of the parameters in the Aggregation Wizard.
3. Click Finish.

Renaming Aggregation Hierarchies

Access the Hierarchies list.

To rename an aggregation hierarchy:

1. Select a hierarchy name from the list and click Rename.
2. Type a new name for the hierarchy, and then press Enter.

Deleting Aggregation Hierarchies

Access the Hierarchies list.

To delete an aggregation hierarchy:

1. Select a hierarchy name from the list and then click Delete.
2. Click Yes.

Note. You cannot delete an aggregation hierarchy that is currently open in the Design Studio.

CHAPTER 14

Working With Weighting Categories

This chapter provides an overview of weighting categories and discusses how to manage weighting categories.

Understanding Weighting Categories

A weighting category consists of a subset of demand points that are associated with a forecast version. A weighting is specified for this subset of forecast data and indicates the importance and reliability of the forecast data during the reconciliation process.

During the forecasting process, collaborative partners provide forecast data for their individual parts of the business. Each demand point in the forecast data must belong to a weighting category. Weighting categories enable forecasters to group forecast data according to type and accuracy. If a partner provides statistical forecast data that historically had reliable accuracy for established product lines, but poor accuracy for new products, that partner's forecast data will be weighted accordingly.

You can assign weighting categories to a demand point set. You do so to manage the importance that specific demand point data has in the reconciliation process. Demand point data in a forecast version can have only one weighting category assigned. To access weighting categories and associated demand point data, a forecast version must already have permissions set.

All demand points are assigned a default weighting category called Default. After you create and assign weighting categories, you can change the weighting category of the demand points.

The system administrator can use these two methods to set weighting categories:

- Design Studio

  Select a subset of the demand model, including forecast version, aggregation hierarchy, and demand points. Then the system administrator creates a weighting category and applies a weight to the forecast data. For example, a demand model can be divided into weighting categories based on product groupings, regional alignments, or individual employee responsibilities.

- DASH commands

  Use the `calculate_weights` command to set a weighting for a specific forecast version in a model. Then, the `import_reconcile_weights` command can be used to import the weighting into the demand model.

Forecasters can use weighting categories in the Consensus Conference Room reconciliation pages. After the weighting categories have been created, you can use the Consensus Conference Room's Reconciliation Wizard to access them in the Consensus Conference Room. Users that have access to multiple forecast versions and the associated demand point data can change the weightings before generating a reconciliation report.

You should also remember that the weight is a relative value that the system compares to other forecast version weightings for the same demand point data. The default value used by the Design Studio is 100.0.

See Also

Chapter 17, "Using the Demand Automation Shell," Calculating Reconciliation Weights, page 174

Chapter 17, "Using the Demand Automation Shell," Import Reconciliation Weights, page 175

Reconciliation Management, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide*

Managing Weighting Categories

This section discusses how to:

- View weighting categories.
- Add weighting categories.
- Edit weighting categories.
- Delete weighting categories.
- Assign weighting categories.
- Change assigned weighting categories.

Windows Used to Manage Weighting Categories

Form Name	Navigation	Usage
Design Studio	Edit, Weighting Categories, Manage Weighting Categories	Displays the Manage Weighting Categories window.
Design Studio	Edit, Weighting Categories, Assign Weighting Categories	Displays the Assign Weighting Categories dialog box.

Viewing Weighting Categories

Access the Manage Weighting Categories window.

To view weighting categories, access the Manage Weighting Categories window.

Adding Weighting Categories

Access the Manage Weighting Categories window.

To add a weighting category:

1. Click Add.
2. Complete these fields:
   - Category Name
   - Weight

3. Click OK.

Editing Weighting Categories

Access the Manage Weighting Categories window.

To edit a weighting category:

1. Select a category name and click Edit.
2. Revise these fields as necessary:
   - Category Name
   - Weight
3. Click OK.

Deleting Weighting Categories

Access the Manage Weighting Categories window.

To delete a weighting category:

1. Select a weighting category name and click Delete.
2. Click OK.

Assigning Weighting Categories

Access the Assign Weighting Categories dialog box.

To assign weighting categories:

1. Complete these fields:
   - Forecast Version
   - Aggregation Hierarchy
   - Demand Point Set
2. Click Select.

Changing Assigned Weighting Categories

Access the Assign Weighting Categories dialog box.

To change an assigned category:

1. Complete these fields:
   - Forecast Version
   - Aggregation Hierarchy
   - Demand Point Set
2. Click Select.
3. Click Change Category.

4. Select the category that you want to apply to the highlighted demand points from the available category options.
5. Click OK.

CHAPTER 15

Working With Weighting Category Accuracy Reports

This chapter provides an overview of weighting category accuracy reports and discusses how to manage weighting category accuracy reports.

Understanding Weighting Category Accuracy Reports

Assigning weighting categories to demand point data enables you to calculate and assign a weighting to each partner's forecast data. The weighting category accuracy report enables you to view the weighting categories, the weight assigned to the category, Mean Absolute Percentage Error (MAPE) and the tracking signal value for each weighting category in the model.

The report provides information about the accuracy of the weighting category. If the weighting category displays a tracking signal value that is too high or low, the weighting category value needs to be reevaluated and changed.

You can generate a report, delete a report and export the report to a file. You can export the report as an XML or a CSV file.

Weighting Category Accuracy Report Generation

The weighting category accuracy report displays the weighting categories for demand points. The report displays the weighting category, assigned weight, MAPE and the tracking signal.

The tracking signal is based on the ratio of the cumulative sum of error between forecast history and actuals to the Mean Absolute Deviation (MAD). Different types of measurements are used to determine how the results of a forecast fit within the upper and lower control limits. Consistent negative or positive growth of the tracking signal indicates that the forecasts are trending too low or too high and that they should be reviewed.

The report includes this information:

- Report Name: The date and time that the report was generated.
- Starting Period: The start date used in the report.
- Ending Period: The end date used in the report.
- Weighting Category: The name of the weighting category for the forecast version.
- Weight: The weighting value that is assigned to a forecast version
- MAPE: The mean absolute percentage error for weighting categories in the demand model.
- Tracking Signal: Tracking signals measure whether the forecasting method is biased over time.

  The system displays a list of forecast errors. The best results are values that are close to zero.

Weighting Category Accuracy Reports Export

You can export a weighting category accuracy report file to a location on a computer as either a an XML or a CSV file. Saving the report file to a location enables you to store reports for comparison or as historical data.

You can open the CSV file in an application such as Microsoft Excel. The XML file can be opened in a browser or XML editor. However the file can be converted into an HTML report based on formatting requirements.

Managing Weighting Category Accuracy Reports

This section discusses how to:

- Generate weighting category accuracy reports.
- View weighting category accuracy reports.
- Delete weighting category accuracy reports.
- Export weighting category accuracy reports.

Window Used to Manage Weighting Category Accuracy Reports

Form Name	Navigation	Usage
Design Studio	Edit, Weighting Categories, Accuracy Reports	Displays the Weighting Category Accuracy Reports window.

Generating Weighting Category Accuracy Reports

Access the Weighting Category Accuracy Reports window.

To generate a weighting category accuracy report:

1. Complete these fields:
   - Starting Period
   - Ending Period
2. Click Generate.

Viewing Weighting Category Accuracy Reports

Access the Weighting Category Accuracy Reports window.

To view an weighting category accuracy report, select a report.

The system displays the details associated with the report that you selected.

Deleting Weighting Category Accuracy Reports

Access the Weighting Category Accuracy Reports window.

To delete a weighting category accuracy report:

1. Select the report that you want to delete.
2. Click Delete.
3. Click Yes.

Exporting Weighting Category Accuracy Reports

Access the Weighting Category Accuracy Reports window.

To export weighting categories accuracy reports:

1. Select the report that you want to export.
2. Click Export.
3. Complete these fields:
   - Save In
   - File name
   - Save as type

CHAPTER 16

Working with the User Manager

This chapter provides an overview of the User Manager and discusses how to:

- Start and log into the User Manager.
- Add users.
- Duplicate users.
- View and edit user information.
- Change user passwords.
- Delete users.
- Grant access to forecast versions.
- Grant access to demand point sets.
- Grant access to filters.
- Grant access to queries.
- Revoke permissions to demand point sets.
- Set Excel forecast provider permissions.

Understanding the User Manager

Security features restrict access to the components of the application, as well as to specific forecast versions and demand points. Users must be assigned a user ID and password before they can work with any component of the application. A system administrator assigns IDs and passwords to users in the User Manager.

The User Manager is a graphical user interface that enables a system administrator to:

- Add and delete users.
- Change user passwords.
- Give users permissions to view, edit, and reconcile forecast versions.
- Grant rights to forecast providers who use Excel.
- Give users permissions to apply, write, edit, and create filters.

Users can also access the User Manager to change their personal information and passwords. However, they cannot view or edit the information of other users.

Note. You should not perform tasks in the User Manager when the other components of the application are in use.

Starting and Logging into the User Manager

In Windows, you start the User Manager from the Start menu. In UNIX, you start the User Manager from the command line prompt. On both operating systems, a login dialog box appears when you start the User Manager.

Starting and Logging into the User Manager

To start and log into the User Manager:

1. Do one of the following:

   From the Windows Start menu, select Programs, EnterpriseOne Supply Chain Planning 9.0, Demand Management, User Manager.

   At the UNIX command line prompt, type this command:

   path`/common/start/run_dm_user_manager`

   Where *path* is the directory path (for example, `/opt/scp/9.0/`) where the application is installed.
2. Complete these fields:
   - User ID
   - Password
3. Click Login.

Adding Users

Before users can start any component of the application, they must have a user ID and password, both of which are assigned by the system administrator. To assign this information, the system administrator adds user accounts in the User Manager.

Note. Administrators can also use the DASH commands to add users and set passwords.

After setting up user accounts, the system administrator can grant users the necessary permissions to view and edit forecast versions, access demand point sets in a demand model, and give access to the Design Studio or the Forecast Studio.

Adding Users

Access the Add User window.

To add users:

1. Complete these fields:
   - User ID
   - Set Password

- Confirm Password

Note. The remaining fields on the dialog box are optional.

2. Click OK to save the information.
3. If the user is requires access to all components of the application, click the Grant Administrator Privileges option.

   The system adds a button next to the user's ID to indicate that the user has administrator privileges.

Duplicating Users

You can associate a new user profile with the pattern of an existing user by duplicating the existing information and using it for the newly created user profile.

Duplicating Users

Select the ID of the user whose information you want to duplicate, and then access the Duplicate Users window.

To duplicate users:

1. Complete these fields:
   - User ID
   - Set Password
   - Confirm Password

   Note. The remaining fields are optional.
2. Click OK.

Viewing and Editing User Information

System administrators can view and change the information for individual users within the User Manager. Administrators can also view and change their own information. Unless you are logged on as the system administrator, you cannot view or edit information about other users.

Note. If you are not the system administrator, you can only change your own information.

Viewing and Editing User Information

Select the user whose information you want to view and edit, and then access the User Properties window.

To view and edit user information:

1. Revise any of these fields:
   - E-mail Address

- Description
- Full Name
- Telephone
- Mobile
- Fax

2. If the user requires access to all components of the application, including the Design Studio, click the Grant Administrator Privileges option.

   The system adds a button next to the user's ID to indicate that the user has administrator privileges.

3. Click OK.

Changing User Passwords

You should periodically change passwords to ensure that the system remains secure. For security reasons, you should change the system administrator password after you first log into the User Manager.

Note. Users can also change their passwords on the My Account page of the Consensus Conference Room. See the *EnterpriseOne Supply Chain Planning Demand Management 9.0 Consensus Conference Room Implementation Guide* for more information.

Changing User Passwords

Select the ID of the user whose password you want to change, and then access the User Password window.

To change user passwords:

1. Complete these fields:
   - Set Password
   - Confirm Password
2. Click OK.

Deleting Users

You can delete user accounts from within the User Manager. You can also use a DASH command to delete users.

To delete users:

1. Select the ID of the user whom you want to delete.
2. Select Edit, User, Delete.
3. Click Yes.

Granting Access to Forecast Versions

Access permissions control whether a user has the ability to view and edit forecast version data in a particular demand model. You can edit access permissions to grant users more or less access to the data in a particular forecast version. The access permissions that you assign to a user influence which forecast versions the user can view, edit, and reconcile in the Forecast Studio and the CCR.

To grant access to forecast versions:

1. Select the ID of the user whose permissions you want to grant access.
2. Select the demand model that contains the forecast versions to which you want to apply permissions from available options in the Model box.
3. Click Forecast Versions.
4. Select Edit, Permissions, Forecast Versions.
5. Select or clear these options for each applicable forecast version:
   - Read
   - Write
   - Reconcile

   Note. Users who add and edit forecast data need these permissions:

   Write and reconcile permissions
6. Click OK.
7. Click Forecast Versions to view the permissions for forecast versions for the current demand model.

Granting Access to Demand Point Sets

Before users can view demand point sets in the Forecast Studio or in the Consensus Conference Room, they must first be granted the rights to access those demand point sets. Access permissions enable users to view and edit specific forecast versions and associated demand point sets. Users are only able to access the demand points and forecast versions to which they have permissions. Likewise, users can import and export information to and from only the demand point sets to which they have access.

When users generate exception reports in the Consensus Conference Room, their access permissions filter out all demand points and forecast versions that they do not have permission to view.

To grant access to demand point sets:

1. Select the user ID to which you want to grant access to a demand point set.
2. Select the demand model containing the forecast versions to which you want to apply permissions from available options in the Model box.
3. If the user will be working with a specific aggregation hierarchy in conjunction with the demand model, select the applicable hierarchy.

   Otherwise, accept the default value.

4. Select Edit, Permissions, Demand Point Sets.
5. Select the demand point set for which you want to grant access.
6. Select one of these options and then click Next:
   - Add Selected.
   - Add All.
7. Select the demand point set for which you want to grant permissions:
8. Select one or both of these options for each forecast version in the demand point set:
   - Read
   - Write
9. Click Back to repeat steps 6 to 8 for each demand point set that you specified in step 5.
10. Click Finish.

Granting Access to Filters

In the Forecast Studio and the Consensus Conference Room, filters can be applied to the product, location, and channel options. These filters can be applied to any attribute for a particular dimension and are user-specific. Before users can apply, edit, or create filters in the Forecast Studio or in the Consensus Conference Room, they must first be granted the appropriate rights in the User Manager. Only an administrator can grant users the right to be able to apply, edit, or create filters, and these rights apply to specific filters, not all existing filters. As a default value, a user who creates a filter has read and write access to that filter.

There are three different types of permissions, create, read, and write:

Permission	Rights
Create	The right to create a filter. A user who creates a filter is automatically granted the right to apply, edit, rename, delete, and copy that filter.
Read	The right to apply filters. If a user has write permission, they are automatically given read permission.
Write	The right to edit, rename, delete, and copy a filter. When a user is granted write permission, they are automatically given read permission.

To grant a user filter create permission:

1. Click the Filter Permissions button.
2. In the Set User Creation Permissions window, select the Create checkbox to grant permissions to users.
3. Click OK.

To grant a user read permission:

1. In the Filters window, select a filter.
2. Click the Filter Permissions button.
3. In the Filter Permissions window, select the Read checkbox to grant read permissions to users.

To grant a user write permission:

1. In the Filters window, select a filter.
2. Click the Filter Permissions button.
3. In the Filter Permissions window, select the Write checkbox to grant write permissions to users.

   A user who is granted write permission to a filter is also granted read permission to that filter.

Granting Access to Queries

Ad hoc queries allow users search the database for items using any defined criteria. Searches are based on individual properties that exist in the demand model but that might not exist together in any single aggregation hierarchy. Queries are created, viewed, edited, and executed in the Forecast Studio and can be viewed and executed in the Consensus Conference Room.

Access is granted on a query-by-query basis.A user who creates a query is automatically granted the right to execute, edit, rename, delete, and copy that query.

Permission	Rights
Read	The right to view the query. A user who creates a query is automatically granted the right to execute, edit, rename, delete, and copy that query.
Execute	The right to execute the query.

To grant access to a query:

1. In the User Manager, select the user to which you want to grant read or execute access.
2. In the Queries window, click the Query Permissions button.
3. Select the query to which you want to add or remove privileges and then click Query Permissions.
4. In the Query Permissions window, do one or more of the following:
   - Click the Read checkbox to allow the specified user to view the Query.
   - Click the Execute checkbox to grant execute permissions to allow the specified user to run the Query.
5. Click OK.

Revoking Permissions to Demand Point Sets

You can revoke a user's permissions to demand point sets. For example, you might revoke permissions for these reasons:

- The user's roles and responsibilities have changed.
- The user was mistakenly given permissions to a demand point set.

Note. Only the system administrator can revoke a user's permissions to demand point sets.

To revoke access to demand point sets:

1. From the list of user IDs, select the ID of the user for whom you want to revoke access to a demand point set.
2. Select the demand model that contains the forecast versions to which you want to apply permissions from available options in the Model box.
3. If the user will be working with a specific aggregation hierarchy in conjunction with the demand model, select the applicable hierarchy. Otherwise, accept the default value.
4. Select Edit, Permissions, Demand Point Sets.
5. Select the demand point set to which you want to revoke access.
6. Select one of these options and then click Next:
   - Remove Selected.
   - Remove All.
7. Click Yes.

Setting Excel Forecast Provider Permissions

Forecast providers are those use the Microsoft Excel spreadsheet application to supply the application with forecast data. Setting Excel forecast provider permissions enables users to access only the forecast versions and demand point sets that they need to view or edit.

System administrators can grant forecast providers access to any or all of the forecast versions and demand point sets in a demand model. You can do this regardless of whether you set up the demand point sets in the Design Studio or by using the scenario generation functions in the Forecast Studio.

To set Excel forecast provider permissions, you must grant the user access to:

- Forecast versions for all hierarchies.
- Demand point sets - one demand point set for each hierarchy.

Note. The Excel workbook is accessible from the Resources tab of the Consensus Conference Room. The system administrator can send the Excel workbook by email to users who do not have access to the CCR.

To grant forecast providers access to forecast versions:

1. From the list of user IDs, select the ID of the user to whom you want to grant access to forecast versions.
2. Select the demand model that contains the forecast versions to which you want to apply permissions from available options in the Model box.
3. From available options in the Aggregation Hierarchy box, select the aggregation hierarchy that is used in conjunction with the demand model.

   The default value is *All*.

   Note. You can grant forecast providers access to forecast versions when the aggregation hierarchy except when the hierarchy is set to *All.*
4. Select Edit, Permissions, Forecast Versions.
5. For each applicable forecast version, select or clear these options:

- Download
- Upload

To select all of the previous options, click Select All. To clear all of the options, click Clear All.

6. Click OK.

To grant forecast providers access to demand point sets:

1. From the list of User IDs, select the user to whom you want to grant access to forecast versions.
2. Complete the Model field by choosing the demand model that contains the forecast versions to which you want to apply permissions.
3. Complete the Aggregation Hierarchy field.
4. Select Edit, Permissions, Forecast Provider Demand Point Sets.
5. Complete the Demand Point Set field.
6. Click OK.
7. Repeat steps 3 to 6 for each hierarchy to which you want to grant the user access.

CHAPTER 17

Using the Demand Automation Shell

This chapter provides an overview of Demand Automation Shell and discusses how to:

- Log in to the Demand Automation Shell.
- Use Tcl to execute Demand Automation Shell commands.
- Get help.
- Use Demand Automation Shell command categories.

Understanding the Demand Automation Shell

The Demand Automation Shell (DASH) is a command line tool that enables users to automate EnterpriseOne Demand Management and the EnterpriseOne Demand Forecasting tasks. DASH is a Tcl (Tool Command Language) shell where scripts where you can execute a script that contains a single command or that contains multiple commands. You can use scripts that are bundled with DASH or create customized scripts using Tcl.

DASH commands allow for batch processing, which enables ease of integration by automating specific functions. For example, setting permissions for multiple users can be time-consuming. Using multiple Demand Automation Shell commands, you can execute one script that automatically sets the permissions for all users.

DASH also provides tools to assist with automation of tasks associated with administration tasks such as environment setup and forecast data administration.

After the application is installed, you can access several commands without logging into DASH. This table describes several commands that are available before you log in to the system:

Command	Action
`login -dc -df`	Logs you into the application that you specified. Specifying `-dc` logs you into Demand Management. Specifying `-df` logs you into Demand Forecasting.
`help`	Prints a list of commands that you can use prior to logging in to DASH.
`list_locks`	Returns a list of string IDs in which each ID identifies a lock.
`delete_locks`	Takes a user-provided List of lock IDs and delete all valid locks that the command can find.
`version`	Displays the version of the application that is installed.
`create_database`	Creates the database. This command must be run as part of the installation process.

Before you can access all of the DASH commands, you must have the following available:

- Database connection.
- A valid login name and password set up by the system administrator.
- A valid license for Demand Management, Demand Forecasting, or both.

Logging In to the Demand Automation Shell

To access DASH commands, you must log in to the specific application for which you want to initiate commands. Only those commands that you have permissions to use are available after you log in.

The administrative user cannot be deleted or renamed. When creating the database using Demand Automation Shell commands, the system creates the administration user and assigns a default administration password. This password can be changed either by using Demand Automation Shell commands or with the User Manager application.

To log in to DASH:

1. Navigate to the <path>\scp\9.0\common\start directory, where <path> is the drive where you installed Demand Management.
2. Enter one of these commands:
   - In Windows, run the batch file `run_dm_dash.bat`.
   - In UNIX, run the script `run_dm_dash`.
3. From the command line, enter the following command:

   `dm::login [-`*dc*`][-`*df*`]` *username password*

   `[-dc]` is Demand Consensus and `[-df]` is Demand Forecasting. You can log in to one application or log in to both applications. For example, to log in to Demand Consensus as Bradley, type this:

   ```
   dm::login -dc Bradley Bradley
   ```

Using Tcl to Execute Demand Automation Shell Commands

Tcl (Tool command language) is a scripting language that you use with DASH. Using a combination of Tcl and Demand Automation Shell commands, you can control, extend, and customize functions within the application.

You can use Tcl to write procedures that are specific to the requirements. For example, you can write a Tcl script to set up a demand model and add forecast versions to the model, or to export forecasts for use in reports.

You must understand these basic programming concepts when using Tcl scripts with DASH:

- Transactions
- Namespace
- Exceptions
- Commits and Rollbacks

Transactions

Both EnterpriseOne Demand Consensus and EnterpriseOne Demand Forecasting use an object oriented database to persist all application data. Changes to the application data are made through transactions. A transaction can hold several Demand Automation Shell commands. The commands in a transaction are carried out in sequential order. You must be logged into DASH before you can use the transaction command.

If a transaction to a database succeeds, the changes made during the transaction are written to the database. If the transaction fails, no changes are made to the database. The syntax of the transaction command is:

```
dm::transaction
{<command1>

<command2>

?

<command1>

<command2>

?}
```

Example

```
dm::transaction {

set cur_date [clock format [clock seconds] -format %Y-%m-%d]

set models [dm::list_models]

foreach model $models {

dm::copy_model $model $model.$cur_date

 }

}
```

In the previous example, the command makes a copy (backup) of all of the models that are based on the current date.

Note. An actual backup copy should be created with XML exports and stored off-site, or, at a minimum, on another computer.

See Also

Chapter 17, "Using the Demand Automation Shell," Commits and Rollbacks, page 100

Namespace

All commands for the application are stored in the namespace. The namespace uniquely identifies a set of related command and variable names.

You can avoid having to prefix all DASH commands with `dm::` by using the `namespace` command. This transaction is valid for using the`namespace`command:

```
namespace import [-force] ::dm::*

transaction
```



```
{set cur_date [clock format [clock seconds] -format %Y-%m-%d]
set models [list_models]
foreach model $models {
copy_model $model $model.$cur_date}}
```

Exceptions

All Demand Automation Shell commands throw a usage exception if you specify an invalid number of arguments. DASH does not commit the transaction if one of the DASH functions throws an exception and that exception is not caught inside the transaction. However, DASH commits the transaction if a portion of the transaction was already committed with a direct commit. DASH commands can also throw other types of exceptions, the causes of which vary with each command.

Commits and Rollbacks

The command commits any changes that were made up to that point within the transaction. You can use this command in conjunction with lengthy transactions when certain points in the transactions are reached and performing a partial commit is safe.

You can also explicitly call the rollback command within a transaction to roll back any changes made up to that point within the transaction that are not committed. If, for example, a lengthy transaction is carried out and some parts of the transaction fail, you can still proceed with the transaction. Using the rollback command, you can roll back only the changes that have been made up to the point of the failure. This is illustrated in this code sample:

```
dm::transaction {
import_business_model ModelBikes Model.xml
commit
catch
{import_forecast_version Model Sales Sales.xml
import_forecast_version Model Marketing Marketing.xml
import_forecast_version Model Statistical Statistical.xml}
exception
if { $exception != "" }
{rollback}}
```

The business model is set from the Model.xml file and then directly committed to the database. The script then imports several forecast files. If any import fails, then all of the import commands are rolled back such that either three forecast versions are populated with forecast data or none of the forecast versions are populated with data.

Note. Demand Automation Shell commands make changes to the data model. Those changes are not written to the actual database until the transaction ends or is committed.

If you use the rollback command, all of the changes since the last commit that you made to the data model are ignored in the transaction. The rollback does not affect or undo any regular Tcl variables.

Getting Help

The `help` command provides information about single or multiple Demand Automation Shell commands that are currently enabled. After you log in, the list of Demand Automation Shell commands that appears depends on the parameters that you provided when you logged in.

Usage

```
help [-list] [command]
```

Parameters

The `help` command takes these parameters:

Reference	Description
`list`	Displays the usage for that command to the screen as a list.
`command`	Displays the usage for the specific command.

Example 1

From the command prompt, type:

```
dm::help
```

When run without parameters, the help command displays a list of all DASH commands.

Example 2

From the command prompt, type:

```
dm::help [-list] [command]
```

The combination of `help`,`-list`, and the name of a Demand Automation Shell command displays the usage for that command. For example, if you specify `list_locks` as the command, the usage is displayed like this:

```
list_locks [-before YYYY-MM-DD]
```

Example 3

From the command prompt, type:

```
dm::help [command]
```

The combination of help and the command displays the usage for the specific command.

Using Demand Automation Shell Command Categories

DASH commands are grouped into these categories:

- Database Locking
- Database Management
- Publish Profile Management

- Navigation Management
- Demand Model Management
- Aggregation Hierarchy Management
- Business Model Management
- Exception Report Management
- Forecast Version Management
- Effective Dating Management
- Scenario Data Management
- Batch Queue Management
- Reconciliation Management
- Sales History Data Management
- Historical Forecast Management
- Historical Series Management
- User Management
- Access Rights Management
- Accuracy Report
- Demand Point Sets Management

Database Locking

This section provides an overview for database locking commands and discusses how to:

- List database locks.
- Delete database locks.

Understanding Database Locking Commands

These commands are used to clear any valid locks that are held by the database:

- `list_locks`
- `delete_locks`
- Any user can execute the database locking commands. Do not execute the database locking commands when an open application shares the same database in which the locks are being deleted.

Listing Database Locks

The `list_locks` command returns a list of string IDs in which each ID identifies a database lock. When this command is used without any argument, it lists all locks in the database. If the `-before` option is used, this command lists only the locks before the specified date.

Usage

```
list_locks [-before YYYY-MM-DD]
```

Parameters

The `list_locks` command takes this parameter:

Parameter	Description
`[-before YYYY-MM-DD]`	List the open locks before the specified date.

Example

From the command prompt, type:

```
dm::list_locks [-before 2002-06-01]
```

The `list_locks` command returns this value:

8-0-3092?5342992 8-0-178560?534299

Deleting Database Locks

The `delete_locks` command takes a list of lock IDs as an argument and deletes all valid database locks that it can find. A valid ID for a lock contains a plus sign (for example, 8-0-3092?7901254). DASH displays an error message if a lock ID does not contain the plus sign.

Usage

```
delete_locks [lock list]
```

Parameters

The `delete_locks` command takes this parameter:

Parameter	Description
`[lock list]`	A list of valid locks for deletion.

Example

From the command prompt, type:

```
dm::delete_locks 8-0-3092?5342992 8-0-3092?7901254
```

Database Management

This section provides an overview of database management commands and discusses how to:

- Create a database.
- Carry out transactions.
- Test a database.

- Delete a database.

Understanding Database Management Commands

These database management commands are available in DASH:

- `create_database`
- `transaction`
- `test_db`
- `delete_database`

Creating Databases

The `create_database` command is enabled before login and must be run once as part of the installation process for the application. The command fails if the database already exists.

When you create the database using DASH, the system always creates the administrator user with the default administration password.

Usage

```
create_database schema_file_path1 [schema_file_path2?]
```

Parameters

The `create_database` command takes this parameter:

Parameter	Description
`schema_file_path1`	The full path name of the schema files to be used. The path must be specified in a UNIX forward slash style.

Example

From the command prompt, type:

```
dm::create_database schema.sch
```

Carrying Out Transactions

All application data is stored in an object-oriented database. Any changes to the application data are made using transactions. Transaction commands can only be run after logging into DASH.

Usage

```
transaction body
```

Parameters

The transaction command take this parameter:

Reference	Description
`body`	A list of Tcl commands

Example

From the command prompt, type:

```
dm::transaction
{import_business_model Model Bikes Model.xml
commit catch
{import_version Model Sales Sales.xml
import_version Model Marketing Marketing.xml
import_version Model Statistical Statistical.xml}
exception
if { $exception != "" }
{rollback}}
```

Testing Databases

The `test_db` command verifies that the database was successfully created during installation.

Note. If the database does not exist, the `test_db` command automatically creates it for you.

Usage

`test_db [-verbose]`

Parameters

The`test_db` command takes this parameter:

Parameter	Description
`[-verbose]`	Specifies whether or not to include detailed information about the database in the report.

Example

From the command prompt, type:

`dm::test_db`

Deleting Databases

The `delete_database` command removes the object-oriented database. You can use this command to delete a corrupt database, or to remove application data before upgrading to a newer version of the application. This command has no parameters.

Usage

`delete_database`

Example

From the command prompt, type:

```
dm::delete_database
```


Publish Profile Management

This section provides an overview of publish profile management commands and discusses how to:

- Import publish profiles.
- Export publish profiles.
- List publish profiles.
- Execute publish profiles.
- Delete publish profiles.
- Rename publish profiles.

Understanding Publish Profile Management Commands

These publish profile management commands are available in DASH:

- `import_publishprofiles`
- `export_publishprofiles`
- `list_publishprofiles`
- `execute_publishprofile`
- `delete_publishprofile`
- `rename_publishprofile`

Importing Publish Profiles

The `import_publishprofiles` command imports publish profiles into a demand model.

Usage

`import_publishprofiles`

Parameters

The `import_publishprofiles` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which the publish profiles are imported.
`filename`	The name of the publish profile to import into the demand model.

Example

At the command prompt, type:

```
dm::transaction
 {import_publishprofiles Bikes BikesProfiles.xml}
```

Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Listing Publish Profiles

The `list_publishprofile` command lists publish profiles that exist in a demand model.

Usage

`list_publishprofile`

Parameters

The `list_publishprofile` command takes this parameter:

Parameter	Description
`model`	The name of the demand model from which the list of publish profiles is generated.

Example

At the command prompt, type:

```
dm::transaction
 {list_publishprofiles Bikes}
```

Exceptions

This command throws an exception if the demand model does not exist.

Executing Publish Profiles

The `execute_publishprofile` command executes an existing publish profile in a demand model.

Usage

`execute_publishprofile`

Parameters

The `export_publishprofile` command takes these parameters:

Parameter	Description
`model`	The name of the demand model in which the publish profile is executed.
`publish_profile`	The name of the publish profile that is executed.

Example

At the command prompt, type:

```
dm::transaction
 {execute_publishprofiles Bikes SalesProfile}
```


Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Deleting Publish Profiles

The `delete_publishprofile` command deletes a publish profile from a demand model.

Usage

`delete_publishprofile`

Parameters

The `delete_publishprofile` command takes these parameters:

Parameter	Description
`model`	The name of the demand model which contains the publish profile that you want to delete.
`publish_profile`	The name of the publish profile that you want to delete.

Example

At the command prompt, type:

```
dm::transaction
 {delete_publishprofiles Bikes SalesProfile}
```

Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Renaming Publish Profiles

The `rename_publishprofile` command renames an existing publish profile from a demand model.

Usage

`rename_publishprofile`

Parameters

The `rename_publishprofile` command takes these parameters:

Parameter	Description
`model`	The name of the demand model which contains the publish profile that you want to rename.
`from_publish_profile`	The name of the publish profile that you want to rename.
`to_publish_profile`	The new name that you want to assign to the publish profile.

Example

At the command prompt, type:

```
dm::transaction
 {rename_publishprofiles Bikes SalesProfile MarketingProfile}
```

Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Navigation Management

This section provides an overview of navigation management commands and discusses how to:

- Export navigation filters.
- Import navigation filters.
- Export navigation filters with access rights.
- Import navigation filters with access rights.
- List navigation filters.
- Delete navigation filters.

Understanding Navigation Management Commands

These navigation profile management commands are available in DASH:

- `export_navigation_filters`
- `import_navigation_filters`
- `export_navigation_filters_access_rights`
- `import_navigation_filters_access_rights`
- `list_navigation_filters`
- `delete_navigation_filter`

Exporting Navigation Filters

The `export_navigation_filters` command exports navigation filters from a demand model.

Usage

`export_navigation_filters`

Parameters

The `export_navigation_filters` command takes these parameters:

Parameter	Description
model	The name of the demand model from which the navigation filters are exported.
filename	The name of the navigation filters to export from the demand model.

Example

At the command prompt, type:

```
dm::transaction
 {export_navigation_filters Bikes BikesFilters.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Navigation Filters

The `import_navigation_filters` command imports navigation filters into an existing demand model.

Usage

`import_navigation_filters`

Parameters

The `import_navigation_filters` command takes these parameters:

Parameter	Description
model	The name of the demand model into which the navigation filters are imported.
filename	The name of the navigation filter that you want to import into the demand model.

Example

At the command prompt, type:

```
dm::transaction
 {import_navigation_filters Bikes BikesFilters.xml}
```

Exceptions

This command throws an exception if the demand model or file name does not exist.

Exporting Navigation Filters With Access Rights

The `export_navigation_filters_access_rights` command exports navigation filters and their associated access rights from a demand model.

Usage

`export_navigation_filters_access_rights`

Parameters

The `export_navigation_filters_access_rights` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the navigation filters and access rights.
`filename`	The name of the file that contains the navigation filters and access rights.

Example

At the command prompt, type:

```
dm::transaction
 {export_navigation_filters_access_rights Bikes BikesFiltersPermissions.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Navigation Filters With Access Rights

The `import_navigation_filters_access_rights` imports navigation filters and their associated access rights into an existing demand model.

Usage

`import_navigation_filters_access_rights`

Parameters

The `import_navigation_filters_access_rights` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import navigation filters and access rights.
`filename`	The name of the file that contains the navigation filters and access rights.

Example

At the command prompt, type:

```
dm::transaction
 {import_navigation_filters_access_rights Bikes BikesFilterPermissions.xml}
```

Exceptions

This command throws an exception if the demand model or the file name does not exist.

Listing Navigation Filters

The `list_navigation_filters` lists navigation filters in a demand model.

Usage

`list_navigation_filters`

Parameters

The `list_navigation_filters` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to generate a list of available navigation filters.

Example

At the command prompt, type:

```
dm::transaction
 {list_navigation_filters Bikes}
```

Exceptions

This command throws an exception if the demand model does not exist.

Deleting Navigation Filters

The `delete_navigation_filter` command deletes a navigation filter.

Usage

`delete_navigation_filter`

Parameters

The `delete_navigation_filter` command takes these parameters:

Parameter	Description
`model`	The name of the demand model which contains the navigation profile that you want to delete.
`navigationFilter`	The name of the navigation filter that you want to delete.

Example

At the command prompt, type:

```
dm::transaction
 {delete_navigation_filters Bikes MountainBikesCanada}
```

Exceptions

This command throws an exception if the demand model or the filter name does not exist.

Demand Model Management

This section provides an overview of demand model management commands and discusses how to:

- Create a new demand model.
- Rename a demand model.
- Copy a demand model.
- Delete a demand model.
- List a demand model.

Understanding Demand Model Management Commands

These Demand Automation Shell commands are used to manage demand models:

- `create_model`
- `rename_model`
- `copy_model`
- `delete_model`
- `list_models`

Creating New Demand Models

The `create_model` command creates a new demand model and sets the available forecast versions for it.

Usage

```
create_model model [forecast_version1 forecast_version2...]
```

Parameters

The `create_model` command takes these parameters:

Parameter	Description
`model`	The name of the demand model to create.
`forecast_version`	The name of the forecast version to create in the demand model.

Example

From the command prompt, type:

```
dm::transaction
 {create_model NewModel Sales Marketing}
```

Exceptions

This command throws an exception if the demand model already exists.

Renaming Demand Models

The `rename_model` command renames an existing demand model.

Usage

```
rename_model from_model to_model
```

Parameters

The `rename_model` command takes these parameters:

Parameter	Description
`from_model`	The name of the demand model to rename.
`to_model`	The new name of the demand model. This model must not exist or the system generates an error.

Example

From the command prompt, type:

```
dm::transaction
 {rename_model Model Model1}
```

Exceptions

This command throws an exception for these conditions:

- A demand model to be renamed does not exist.
- A demand model with the target name already exists.

Copying Demand Models

The `copy_model` command makes a copy of an existing demand model.

Usage

```
copy_model from_model to_model
```

Parameters

The `copy_model` command takes these parameters:

Parameter	Description
`from_model`	The name of the demand model to copy.
`to_model`	The name of the new demand model.

Example

From the command prompt, type:

```
dm::transaction
 {copy_model Model Model1}
```

Exceptions

This command throws an exception for these conditions:

- The demand model to be copied does not exist.
- A target model with the specified name already exists.

Deleting Demand Models

The `delete_model` command deletes an existing demand model.

Usage

```
delete_model model
```

Parameters

The `delete_model` command takes this parameter:

Parameter	Description
`model`	The name of the demand model that you want to delete from the database.

Example

From the command prompt, type:

```
dm::transaction { delete_model Model}
```

Exceptions

This command throws an exception if the demand model does not exist.

Listing Demand Models

The `list_models` command returns a list with the names of all demand models in the database. This command has no parameters.

Usage

```
list_models
```

Example

From the command prompt, type:

```
dm::transaction
{list_models}
```

Aggregation Hierarchy Management

This section provides an overview of aggregation hierarchy management commands and discusses how to:

- Add an aggregation hierarchy.
- Delete an aggregation hierarchy.
- Rename an aggregation hierarchy.
- List aggregation hierarchies.
- Import an aggregation hierarchy.
- Export an aggregation hierarchy.

Understanding Aggregation Hierarchy Management Commands

You can use these commands to manage aggregation hierarchies:

- `add_aggregation_hierarchy`
- `delete_aggregation_hierarchy`
- `rename_aggregation_hierarchy`
- `list_aggregation_hierarchies`
- `import_aggregation_hierarchy`
- `export_aggregation_hierarchy`

Adding Aggregation Hierarchies

The `add_aggregation_hierarchy` command to creates a new aggregation hierarchy.

Usage

```
add_aggregation_hierarchy model aggregation_hierarchy
```

Parameters

The `add_aggregation hierarchy` command takes these parameters:

Parameter	Description
`model`	The name of the demand model to which you want to add an aggregation hierarchy.
`aggregation_hierarchy`	The name of the aggregation hierarchy that you want to add to the demand model.

Example

From the command prompt, type:

```
dm::transaction
 {add_aggregation_hierarchy model ProductGroupA}
```

Exceptions

This command throws an exception for these conditions:

- The aggregation hierarchy with the specified name already exists in the model.
- The demand model does not exist.

Deleting Aggregation Hierarchies

The `delete_aggregation_hierarchy` command deletes an existing demand model.

Usage

```
delete_aggregation_hierarchy model aggregation_hierarchy
```

Parameters

The `delete_aggregation_hierarchy` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to delete an aggregation hierarchy.
`aggregation_hierarchy`	The name of the aggregation hierarchy that you want to delete.

Example

From the command prompt, type:

```
dm::transaction
 {delete_aggregation_hierarchy model ProductGroupA}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist.
- The aggregation hierarchy is the last in the model.

Renaming Aggregation Hierarchies

The `rename_aggregation_hierarchy` command renames an existing aggregation hierarchy.

Usage

```
rename_aggregation_hierarchy model from_name to_name
```

Parameters

The `rename_aggregation_hierarchy` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the aggregation hierarchy that you want to rename.
`from_name`	The name of the aggregation hierarchy to rename.
`to_name`	The new name of the aggregation hierarchy. If this aggregation hierarchy already exists, the system generates an error.

Example

From the command prompt, type:

```
dm::transaction
{rename_aggregation_hierarchy myModel ProductGroupA ProductGroupB}
```

Exceptions

This command throws an exception for these conditions:

- The aggregation hierarchy to be renamed does not exist.
- An aggregation hierarchy with the target name already exists.

Listing Aggregation Hierarchies

The `list_aggregation_hierarchies` command returns a list of names of the existing aggregation hierarchies.

Usage

```
list_aggregation_hierarchies model
```

Parameters

The `list_aggregation_hierarchies` command takes this parameter:

Parameter	Description
`model`	The name of the demand model containing the aggregation hierarchies that you want to list.

Example

From the command prompt, type:

```
dm::transaction
 {list_aggregation_hierarchies model}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Aggregation Hierarchies

The `import_aggregation_hierarchy` command imports an aggregation hierarchy from an XML file.

Usage

```
import_aggregation_hierarchy model aggregation_hierarchy filename
```

Parameters

The `import_aggregation_hierarchy` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import an aggregation hierarchy.
aggregation_hierarchy	The name of the aggregation hierarchy to import.
filename	The name of the XML file containing the aggregation hierarchy that you want to import into the demand model. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_aggregation_hierarchy model Sales ProductGroupB}
```

Exceptions

This command throws an exception for these conditions:

- An aggregation hierarchy to be renamed does not exist.
- The aggregation hierarchy does not exist in the model.
- The import process fails.

Exporting Aggregation Hierarchies

The `export_aggregation_hierarchy` command exports an aggregation hierarchy to an XML file.

Usage

```
export_aggregation_hierarchy model aggregation_hierarchy filename
```

Parameters

The `export_aggregation_hierarchy` command takes these parameters:

Parameter	Description
model	The name of the demand model from which you want to export an aggregation hierarchy.
aggregation_hierarchy	The name of the aggregation hierarchy that you want to export.
filename	The name of the XML file into which you want to export the aggregation hierarchy. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
 {export_aggregation_hierarchy model Sales ProductGroupB}
```

Exceptions

This command throws an exception for these conditions:

- An aggregation hierarchy to be renamed does not exist.
- The aggregation hierarchy does not exist in the model.

Business Model Management

This section provides an overview of business model management commands and discusses how to:

- Import the business model.
- Export the business model.
- Set the horizon for the business model.
- Roll forward the horizon for the model.
- List units of measure for the model.
- Import conversion rates into the demand model.
- Export conversion rates from the demand model.
- Import channels into the demand model.
- Import locations into the demand model.
- Import products into the demand model.
- Import demand points into the demand model.
- Update channels in the demand model.
- Update locations in the demand model.
- Update products in the demand model.
- Clear the New flag for channels.
- Clear the New flag for products.
- Clear the New flag for locations.
- Clear the New flag for demand points.
- Delete channels from the demand model.
- Delete products from the demand model.
- Delete locations from the demand model.
- Delete demand points from the demand model.

Understanding Business Model Management Commands

You can configure the business model using the Design Studio or by using any of the business model management Demand Automation Shell commands. The business model contains the structure of the model. The demand model uses the business model structure to hold the forecast data. The business model contains many different levels of information at different levels of detail. If you use DASH commands to configure the business model, you can use them in any order; no sequential order is necessary.

You can use these commands to manage aggregation hierarchies:

- `import_business_model`
- `export_business_model`
- `horizon`
- `roll_forward`
- `list_units_of_measure`
- `import_conversion_rates`
- `export_conversion_rates`
- `import_channels`
- `import_locations`
- `import_products`
- `import_demand_points`
- `update_channels`
- `update_products`
- `update_locations`
- `delete_channels`
- `delete_products`
- `delete_locations`
- `delete_demand_points`
- `rename_channels`
- `rename_locations`
- `rename_products`

Importing Business Models

The `import_business_model` command imports a business model from an XML file into a demand model.

Usage

```
import_business_model model filename
```

Parameters

The `import_business_model` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import the business model.
filename	The name of the XML file containing the business model that you want to import. When specifying the filename, provide the path where the file resides.

Example

At the command prompt, type:

```
dm::transaction
 {import_business_model Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The import process fails.

Exporting Business Models

The `export_business_model` command exports the business model structure and demand points for a demand model into an XML file. The demand points can be excluded if desired.

Usage

```
export_business_model model filename [-no_leaf_demand_points]
```

Parameters

The `export_business_model` command takes these parameters:

Parameter	Description
model	The name of the demand model for which to export the business model.
filename	The name of the XML file into which you want to export the demand model. When specifying the filename, provide the path where the file should reside.
[-no_leaf_demand_points]	Specifies that demand point information is not exported with the business model.

Example

From the command prompt, type:

```
dm::transaction
 {export_business_model BikesModel BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The export process fails.

Setting the Horizon for Business Models

You can set the horizon for the business model data using the `horizon` command. The way that this command is carried out depends on the number of parameters that are supplied.

Note. You should back up existing models before changing the model horizon . This step is not required for new business models.

Usage

```
horizon model [(yearly|monthly|weekly|daily|fourWeekly|
weekly445|weekly454| weekly544)] YYYY-MM-DD forecast_version_count
sales_history_count
```

If the name of the demand model is the only parameter that is supplied, this command returns the horizon for the demand model in this format:

```
period_type forecast_start_date forecast_count history_count
```

Otherwise the command sets the horizon for the demand model.

Parameters

The `horizon` command uses these parameters:

Parameter	Description
`model`	The name of the demand model with a horizon that you want to change.
`Yearly/monthly/daily/fourWeekly/weekly445/weekly454/weekly544`	The type of time period set for the horizon. This parameter has these restrictions: • Yearly - Cannot start on February 29th. • Monthly - Can only start on days before or on the 28th of the month.
`YYYY-MM-DD`	The start date of all forecasts.
`forecast_version_count`	The number of time buckets that are set for the forecast.
`sales_history_count`	The number of time buckets that are set for the sales history.

Example

To set the horizon to monthly time buckets, starting on June 1, 2002, and with 12 forecast version time buckets and 12 sales history time buckets, type this command at the command prompt:

```
dm::transaction
 {horizon Model monthly 2002-06-02 12 12}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- An invalid type of the period, date value, or date format is specified.
- The `forecast_version_count` parameter has a negative value.
- The `sales history_count` parameter has a negative value.

Rolling Forward the Model Horizon

The `roll_forward` command rolls or moves forecast data forward in time by a specified number of periods. You can roll the horizon forward and use parameters listed below to save forecast history by the specified number of periods before the start of the horizon.

You can also remove forecast history time periods. When the `roll_forward` command is run, the forecast horizon rolls forward adding time periods to both the forecast history and forecast data. For example, if you roll forward the horizon by two month time periods, two additional month time periods are added to the forecast history data. To maintain the forecast history to a specific number of month time periods, use the `[-trim_history]` parameter to remove the additional monthly time periods.

If you roll forward the model with no variables, the system creates empty periods between the old position and the new forward position.

Note. You should back up your model before rolling forward the model horizon.

Usage

```
roll_forward model number_of_periods [-save_history][-trim_history]
```

Parameters

The `roll_forward` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the data to be rolled forward.
`number_of_periods`	The number of time periods to roll the forecast data forward.
`[-save_history]`	Specifies whether to save forecast history in the demand model.
`[-trim_history]`	Removes the same number of time periods specified in the roll_forward command from forecast history.

Example

To shift data from period 2 to period 1, from period 3 to period 2, and so on, type this command at the command prompt:

```
dm::transaction {roll_forward Model 1 -save_history -trim_history}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The number of periods is not an integer greater than or equal to 1.

Listing the Model Units of Measure

The `list_units_of_measure` command returns a list with the names of the units of measure that are defined for the demand model.

Usage

```
list_units_of_measure model
```

Parameters

The `list_units_of_measure` command takes this parameter:

Parameter	Description
`model`	The name of the demand model containing the units of measure that you want to list.

Example

To list the units of measure for the Bikes demand model, type this command at the command prompt:

```
dm::transaction {list_units_of_measure Bikes}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Conversion Rates Into the Demand Model

The `import_conversion_rates` command imports the conversion rates for a unit of measure series into a demand model. The conversion rate information is in an XML file, which specifies the rates for each demand point and each unit of measure in a model. The conversion rate can also be specified as the base unit of measure for each demand point.

The conversion rate information for a unit of measure in a demand point consists of an effective date and the actual conversion rate. This information must appear in chronological order in the XML file.

After a conversion rate is set, it is valid until another conversion rate is set or until the end of the horizon.

Usage

```
import_conversion_rates model filename
```

Parameters

The `import_conversion_rates` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you are importing the conversion rate information.
`filename`	The name of the XML file containing the conversion rate information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_conversion_rates ConferenceRoom BikesModel.xml}
```

Exceptions

This command throws an exception if the demand model contains fewer than two units of measure.

Note. If more than one conversion rate is listed for a time bucket, the application uses the last rate.

If no conversion rate is specified for the first time bucket, the default rate is zero.

If the conversion rate that is provided is after the start of the history horizon, then that rate is used as the default value.

Exporting Conversion Rates from the Demand Model

The `export_conversion_rates` command exports the conversion rates for a unit of measure series from a demand model into an XML file. The base unit of measure for each demand point is also exported to the XML file.

Usage

```
export_conversion_rates model filename
```

Parameters

The `export_conversion_rates` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you are exporting the conversion rate information.
`filename`	The name of the XML file into which you are exporting the conversion rate information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_conversion_rates Model BikesModel.xml}
```

Exceptions

This command throws an exception if the demand model contains fewer than two units of measure.

Importing Channels Into the Demand Model

The `import_channels` command imports channel information from an XML file into a demand model. The source of the channel information is usually an external data source such as an ERP or product data management system.

Usage

```
import_channels model filename
```

Parameters

The `import_channels` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the channel information.
`filename`	The name of the XML file containing the channel information that you want to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_channels Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A channel that is specified in the XML file already exists in the demand model.

Importing Locations Into the Demand Model

The `import_locations` command imports location information from an XML file into a demand model. The source of the location information is usually an external data source such as an ERP or product data management system.

Usage

```
import_locations model filename
```

Parameters

The `import_locations` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the location information.
`filename`	The name of the XML file containing the location information that you want to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
```



```
{import_locations Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A location that is specified in the XML file already exists in the demand model.

Importing Products Into the Demand Model

The `import_products` command imports product information from an XML file into a demand model. The source of the product information is usually an external data source such as an ERP or product data management system.

Usage

```
import_products model filename
```

Parameters

The `import_products` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the product information.
`filename`	The name of the XML file containing the product information that you want to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_products Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A product that is specified in the XML file already exists in the demand model.

Importing Demand Points Into the Demand Model

The `import_demand_points` command imports demand point information from an XML file into a demand model. The source of the demand point information is usually an external data source such as an ERP or product data management system.

Usage

```
import_demand_points model filename
```

Parameters

The `import_demand_points` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the demand point information.
`filename`	The name of the XML file containing the demand point information that you want to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_demand_points Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A demand point that is specified in the XML file already exists in the model.
- An invalid coordinate is specified in the XML file.

Updating Channels in the Demand Model

The `update_channels` command updates the channel information in a demand model using data from an XML file. The source of the channel information is usually an external data source such as an ERP or product data management system.

Usage

```
update_channels model filename
```

Parameters

The `update_channels` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the channels that you want to update.
`filename`	The name of the XML file containing the updated channel information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{update_channels Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A channel that is specified in the XML file does not exist in the demand model.

See Also

Renaming Channels in the Demand Model

Updating Locations in the Demand Model

The `update_locations` command updates the location information in a demand model using data from an XML file. The source of the location information is usually an external data source such as an ERP or product data management system.

Usage

```
update_locations model filename
```

Parameters

The `update_locations` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the locations that you want to update.
`filename`	The name of the XML file containing the updated location information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{update_locations Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A location that is specified in the XML file does not exist in the demand model.

See Also

Renaming Locations in the Demand Model

Updating Products in the Demand Model

The `update_products` command updates the product information in a demand model using data from an XML file. The source of the product information is usually an external data source such as an ERP or product data management system.

Command

`update_products model filename`

Parameters

The `update_products` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the product that you want to update.
`filename`	The name of the XML file containing the updated product information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{update_products Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A product that is specified in the XML file does not exist in the demand model.

See Also

Renaming Locations in the Demand Model

Clearing the New Flag for Channels

The `clear_new_flag_channels` command clears the New flag for channels in a demand model, based on data from an XML file.

Usage

`clear_new_flag_channels model filename`

Parameters

The `clear_new_flag_channels` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the channels with the New flag setting that you want to clear.
`filename`	The name of the XML file containing the list of channels IDs for channels, the New flag settings of which will be cleared. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{clear_new_flag_channels Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A channel ID that is specified in the XML file does not exist in the demand model.

Clearing the New Flag for Products

The `clear_new_flag_products` command clears the New flag for products in a demand model, based on data from an XML file.

Usage

```
clear_new_flag_products model filename
```

Parameters

The `clear_new_flag_products` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the products with New settings you want to clear.
`filename`	The name of the XML file containing the list of product IDs for products, the New flag settings of which will be cleared. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{clear_new_flags_products Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A product ID that is specified in the XML file does not exist in the demand model.

Clearing the New Flag for Locations

The `clear_new_flag_locations` command clears the New flag for locations in a demand model, based on data from an XML file.

Usage

```
clear_new_flag_locations model filename
```

Parameters

The `clear_new_flag_locations` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the locations with a New flag setting that you want to clear.
`filename`	The name of the XML file containing the list of location IDs for locations with New flag settings that will be cleared. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{clear_new_flag_locations Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A location ID that is specified in the XML file does not exist in the demand model.

Clearing the New Flag for Demand Points

The `clear_new_flag_demand_points` command clears the New flag for demand points in a demand model, based on data from an XML file.

Usage

```
clear_new_flag_demand_points model filename
```

Parameters

The `clear_new_flag_demand_points` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the demand points with a New flag setting that you want to clear.
`filename`	The name of the XML file that contains the list of demand point IDs, for which the New flag settings are to be cleared. When specifying the file name, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{clear_new_flag_demand_points Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A demand point ID that is specified in the XML file does not exist in the demand model.

Deleting Channels From the Demand Model

The `delete_channels` command deletes channels in a demand model, based on the channel information contained in an XML file.

Usage

```
delete_channels model filename
```

Parameters

The `delete_channels` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the channels that you want to delete.
`filename`	The name of the XML file containing the channel IDs for channels to be deleted from the demand model. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{delete_channels TotalModel BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A channel ID that is specified in the XML file does not exist in the demand model.

Deleting Products From the Demand Model

The `delete_products model` command deletes products in a demand model, based on the channel information contained in an XML file.

Usage

```
delete_products model filename
```

Parameters

The `delete_products` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the products that you want to delete.
`filename`	The name of the XML file containing the product IDs for products to be deleted from the demand model. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {delete_products Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A product ID that is specified in the XML file does not exist in the demand model.

Deleting Locations From the Demand Model

The `delete_locations` command deletes locations in a demand model, based on the location information contained in an XML file.

Usage

```
delete_locations model filename
```

Parameters

The `delete_locations` command takes these parameters:

Parameters	Description
`model`	The name of the demand model containing the locations that you want to delete.
`filename`	The name of the XML file containing the location IDs for locations to be deleted from the demand model. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{delete_locations Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- A location ID that is specified in the XML file does not exist in the demand model.

Deleting Demand Points From the Demand Model

The `delete_demand_points` command deletes demand point information from a demand model, based on the demand point information in an XML file.

Usage

```
delete_demand_points model filename
```

Parameters

The `delete_demand_points` command takes these parameters:

Parameters	Description
`model`	The name of the demand model in which you want to import the demand point information.
`filename`	The name of the XML file containing the demand point IDs for demand points that you want to delete. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{delete_demand_points Model BikesModel.xml}
```

Exceptions

This command throws an exception for this condition: The demand model does not exist.

Renaming Channels in the Demand Model

The `rename_channels` command renames channels in a demand model using data from an XML file. The source of the channel information is usually an external data source such as an ERP or product data management system.

When using the `rename_channels` command, you cannot rename a coordinate if the name exists elsewhere in the hierarchy.

Usage

```
rename_channels model filename
```

Parameters

The `rename_channels` command takes these parameters:

Parameters	Description
model	The name of the demand model in which you want to rename channels.
filename	The name of the XML file containing the demand points that you want to rename. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{rename_channels Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The channel does not exist in the model.

Renaming Locations in the Demand Model

The `rename_locations` command renames locations in a demand model using data from an XML file. The source of the location information is usually an external data source such as an ERP or product data management system.

When using the `rename_locations` command, you cannot rename a coordinate if the name exists elsewhere in the hierarchy.

Usage

```
rename_locations model filename
```

Parameters

The `rename_locations` command takes these parameters:

Parameters	Description
model	The name of the demand model in which you want to rename locations.
filename	The name of the XML file containing the demand points that you want to rename. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{rename_locations Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The location does not exist in the model.

Renaming Products in the Demand Model

The `rename_products` command renames products in a demand model using data from an XML file. The source of the product information is usually an external data source such as an ERP or product data management system.

When using the `rename_products` command, you cannot rename a coordinate if the name exists elsewhere in the hierarchy.

Usage

```
rename_products model filename
```

Parameters

The `rename_products` command takes these parameters:

Parameters	Description
`model`	The name of the demand model in which you want to rename products.
`filename`	The name of the XML file containing the demand points that you want to rename. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{rename_products Model BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The product does not exist in the model.

Exception Report Management

This section provides an overview of exception report management commands and discusses how to:

- Clear exception reports.
- Export exception reports.
- Import exception reports.

Understanding Exception Report Management Commands

You can use these commands to manage exception reports:

- `clear_exception_reports`
- `export_exception_reports`
- `import_exception_reports`

Clearing Exception Reports

The `clear_exception_reports` command removes the exception report results from the batch queue. The exception report remains in the exception page of the CCR.

Usage

```
clear_exception_reports model
```

Parameters

The `clear_exception_reports` command takes this parameter:

Parameter	Description
`model`	The name of the demand model containing the exception report that you want to clear.

Example

At the command prompt, type:

```
dm::transaction
{clear_exception_reports ConferenceRoom}
```

Exceptions

This command throws an exception if the demand model does not exist.

Exporting Exception Reports

The `export_exception_reports` command exports exception reports from the batch queue to an XML file.

Usage

```
export_exception_reports model filename
```

Parameters

The `export_exception_reports` command takes these parameters:

Parameters	Description
`model`	The name of the demand model containing the exception report that you want to export.

Parameters	Description
hierarchy	The name of the hierarchy.
filename	The name of the XML file to which you want to export the exception report.

Example

From the command prompt, type:

```
dm::transaction {export_exception_reports ConferenceRoom Sales Reports.xml
BikesModel.xml}
```

Exceptions

The command throws an exception if the demand model or the hierarchy do not exist.

Importing Exception Reports

The `import_exception_reports` command imports exception reports into the batch queue from an XML file.

Usage

`import_exception_reports model filename`

Parameters

The `import_exception_reports` command takes these parameters:

Parameters	Description
model	The name of the demand model into which you want to import an exception report.
filename	The name of the XML file from which to import the report data. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_exception_reports ConferenceRoom reports.xml BikesModel.xml}
```

Exceptions

This command throws an exception if the demand model or the file that contains the report data do not exist.

Forecast Version Data Management

This section provides an overview of forecast version data management commands and discusses how to:

- Add new forecast versions.

- Delete forecast versions.
- List forecast versions.
- Set the disaggregation profile.
- Clear forecast versions.
- Disaggregate forecasts.
- Export forecast versions.
- Import forecast versions.
- Delete forecast version annotations.
- Copy forecast version notes into annotations.
- Export multiple forecast versions.
- Import multiple forecast versions.
- Import full forecast versions.
- Export full forecast versions.
- Disaggregate forecasts.
- Balance forecasts.
- Copy forecast versions.
- Copy partial forecast versions.
- Rename forecast versions.
- Sum forecast versions.
- Generate forecast change reports.

Understanding Forecast Version Data Management Commands

You can use these commands to manage forecast versions:

- `add_forecast_version`
- `delete_forecast_version`
- `list_forecast_version`
- `disaggregation_profile`
- `clear_forecast_version`
- `disaggregate`
- `export_forecast_version`
- `import_forecast_version`
- `delete_forecast_version_annotation`
- `copy_forecast_version_notes_into_annotations`
- `export_multiple_forecast_versions`
- `import_multiple_forecast_versions`
- `import_full_forecast_version`

- export_full_forecast_version
- copy_forecast_version
- copy_partial_forecast_version
- rename_forecast_version
- sum_forecast_versions
- report_forecast_change

Adding New Forecast Versions

The add_forecast_version command adds a new forecast version to an existing demand model.

Usage

```
add_forecast_version model forecast_version
```

Parameters

The add_forecast_version command takes these parameters:

Parameter	Description
model	The name of the existing demand model into which you want to add a forecast version.
forecast_version	The name of the forecast version that you want to add to the demand model.

Example

At the command prompt, type:

```
dm::transaction
{add_forecast_version Model Sales}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- Another forecast version with the same name already exists in the demand model.

Deleting Forecast Versions

The delete_forecast_version command deletes a forecast version from an existing demand model.

Usage

```
delete_forecast_version model forecast_version
```

Parameters

The delete_forecast_version command takes these parameters:

Parameter	Description
model	The name of the demand model from which to delete the forecast version.
forecast_version	The name of the forecast version that you want to delete from the demand model.

Example

At the command prompt, type:

```
dm::transaction
 {delete_forecast_version Model Enterprise}
```

Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Listing Forecast Versions

The `list_forecast_versions` command returns a list of the names of forecast versions that are defined in the demand model.

Usage

```
list_forecast_versions model
```

Parameters

The `list_forecast_versions` command takes this parameter:

Parameter	Description
model	The name of the demand model with the forecast versions that you want to list.

Example

At the command prompt, type:

```
dm::transaction {list_forecast_verions Model}
```

Exceptions

This command throws an exception if the demand model does not exist.

Setting the Disaggregation Profile

The `disaggregation_profile` command sets a disaggregation profile. It does this using one of the existing forecast versions in the demand model that you want to model as the disaggregation profile.

Usage

```
disaggregation_profile model forecast_version
```

If the demand model is the only argument supplied, the `disaggregation_profile` command always returns the disaggregation profile for the demand model in this format: *profile_forecast_version*, where *profile_forecast_version* represents the disaggregation profile. If you provide a second argument, it sets the current profile to that argument and then returns the current profile.

Parameters

The `disaggregation_profile` command takes these parameters:

Reference	Description
`model`	The name of the demand model for which you want to set the disaggregation profile.
`forecast_version`	The name of the forecast version that you want to use as the disaggregation profile.

Example

At the command prompt, type:

```
dm::transaction
 {disaggregation_profile Model Sales}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The forecast version does not exist in the demand model.

Clearing Forecast Versions

The `clear_forecast_version` command removes the forecast data contained in a forecast version.

Usage

```
clear_forecast_version model forecast_version
```

Parameters

The `clear_forecast_version` command takes these parameters:

Parameters	Description
`model`	The name of the demand model containing the forecast version that you want to clear.
`forecast_version`	The forecast version from which you want to remove forecast data.

Example

At the command prompt, type:

```
dm::transaction
 {clear_forecast_version Model Sales}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The forecast version does not exist in the demand model.

Disaggregating Forecasts

The `disaggregate` command disaggregates forecasts based on these parameters:

- Forecast version.
- Unit of measure.
- Aggregation level.
- Forecast version profile.

You can also specify whether to leave a demand point forecast in a state that is not balanced above it. If you use the reconciliation process to calculate a consensus forecast, it is recommended that you do not balance above when disaggregating.

Usage

```
disaggregate model aggregation_hierarchy forecast_version unit_of_measure
aggregation_level disaggregation_method [-append_notes] [-balance_above]
```

Parameters

The disaggregate command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the forecast that you want to disaggregate.
`aggregation_hierarchy`	The name of the aggregation hierarchy in the demand model containing the data that you want to disaggregate.
`forecast_version`	The name of the forecast version that you want to use as the disaggregation profile.
`unit_of_measure`	The default unit of measure that is used.
`aggregation_level`	The name of the aggregation level in the demand model on which the disaggregation is based.

Parameter	Description
`disaggregation_method`	The dissagregation method. Options are: • `-system_profile` Disaggregates the forecast based on a system-defined pattern. You can use this option if little or no data is at the lowest level of the current forecast version. Otherwise, the underlying pattern in the forecast version is used to disaggregates the forecast. • `-top_down` Distributes the forecast data downwards from the highest aggregation level according to the underlying pattern in the current forecast version. The value in each time bucket adds up to the value of the parent forecast. • `-bottom_up` Distributes forecast data upwards from the lowest aggregation level according to the underlying pattern in the current forecast version. Leaf-level demand points are copied from the system profile into the working profile. The values that were originally in the working profile at the leaf level are ignored.
`-append_notes`	An optional parameter that automatically appends notes to forecasts during the disaggregation process. **Note.** This parameter is only available when using the `-top_down` disaggregation method.
`-balance_above`	An optional parameter that aggregates above the disaggregated demand point(s). If you use the reconciliation process, it is recommended that you not use this parameter.

Example

From the command prompt, type:

```
dm::transaction
 {disaggregate Bikes "Product Group by Region" Sales Each
  "Product by Region" -system_profile}
```

Exceptions

This command throws an exception if any of the following do not exist:

- Demand model.
- Hierarchy.
- Forecast version.
- Aggregation level.

Exporting Forecast Versions

When consensus is reached and an enterprise forecast is generated, the data can be exported from the application to an external system. Use the `export_forecast_version` command to export the data. This command exports all of the forecast data, including the unit of measure and any missing demand points, into an XML file that contains all of the aggregation points.

Sometimes, external suppliers or partners want to view and edit their own forecast data from the Consensus Conference Room (CCR). Because they do not have direct access to the CCR, a system administrator must extract and send the forecast data to them. A system administrator can extract specific forecast data from the CCR by using the `export_forecast_version` command and send the file to the external supplier or partner.

The `export_forecast_version` command exports data into an XML file. This XML file conforms to the document definition file that is contained in the document type definition (DTD) file named Export-Parm.dtd as follows:

```
<!-- Document hierarchical structure -->

<!ELEMENT export-parm ( slice-list )>

<!ELEMENT slice-list ( slice* )>

<!ELEMENT slice ( pyramid-point, depth )>

<!ELEMENT pyramid-point ( channel-path, location-path, product-path )>

<!-- Document value elements -->

<!ELEMENT depth ( #PCDATA )>

<!ELEMENT channel-path ( #PCDATA )>

<!ELEMENT location-path ( #PCDATA )>

<!ELEMENT product-path ( #PCDATA )>
```

A file named slices_file.xml contains a list of all the business model points (aggregation points) that are exported to an external system. This file indicates the specific demand points that are captured and exported. For example, the slices_file.xml file can contain these attributes:

```
<slice-list>

<slice>

<aggregation-point>

<channel-path>/</channel-path>

<location-path>/Europe/Italy/Roma</location-path>

<product-path>/Pacemakers/Implant</product-path>

</aggregation-point>

<depth>1</depth>

</slice>

</slice-list>
```


You can specify whether to export aggregated values (non-leaf) or missing values. The `export_forecast_version` command is useful when you want to export the forecasts from Demand Management into another application that cannot aggregate forecasts.

Usage

```
export_forecast_version model forecast_version unit_of_measure filename
[-aggregates] [-missing] [-annotations][slices_file.xml]
```

Parameters

The `export_forecast_version` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export a forecast version.
`forecast_version`	The name of the forecast version that you want to export.
`unit_of_measure`	The unit of measure in which the forecast data is exported.
`filename`	The name of the file from which you want to export data.
`[-aggregates]`	Aggregated (non-leaf) demand points are exported.
`[-missing]`	Aggregation points that are missing but will be exported.
`[-annotation]`	Parameter specifying that annotations are exported.
`[slices_file.xml]`	The file containing the aggregation point information that is exported.

Example

At the command prompt, type:

```
dm::transaction
{export_forecast_version Model Sales dollars forecasts.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The forecast version does not exist in the demand model.
- The unit of measure does not exist in the demand model.

Importing Forecast Versions

A newly created demand model does not contain any forecast data. You can upload and download individual forecast data into it by using the ForecastWeb Microsoft Excel add-in, or you can use DASH to import multiple forecast data at once.

The `import_forecast_version` command imports forecasts into a forecast version in the demand model. At any particular point, existing forecasts are either overridden by the new data if the import contains a forecast for that point, or they are left unaltered if the imported data does not include data for that point.

Usage

```
import_forecast_version model aggregation_hierarchy forecast_version
filename [-disaggregate_profile|-disaggregate_top_down|
-disaggregate_bottom_up] [-balance_above]
```

Parameters

The `import_forecast_version` command takes these parameters:

Reference	Description
`model`	The name of the demand model into which forecasts are imported.
`aggregation_hierarchy`	The name of the aggregation hierarchy into which you want to import the forecast version data.
`forecast_version`	The name of the forecast version into which you want to import the data.
`filename`	The name of the file from which data is imported.
`-disaggregate_profile`	A flag that causes the`import_forecast_version` command to distribute the forecast data downward, based on a system-defined pattern.
`-top_down`	A flag that disaggregates forecast data from the aggregation level that you specify down to the lowest level of aggregation.
`-bottom_up`	A flag that disaggregates forecast data from the aggregation level that you specify up to the highest level of aggregation.
`–balance_above`	An optional parameter that aggregates above the disaggregated demand point(s). If you use the reconciliation process, it is recommended that you not use this parameter.

Example

At the command prompt, type:

```
dm::transaction
 {import_forecast_version Model Sales1 forecasts.xml -disaggregate_top_down}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist in the demand model.
- An invalid file name or format is used.
- The disaggregation flag is invalid.

Deleting Forecast Version Annotations

The `delete_forecast_version_annotations` command deletes the annotations in a specific forecast version. You can delete all the annotations in the forecast version, or only the annotations that were entered before a certain date.

Usage

```
delete_forecast_version_annotations model forecast_version -all | -before
YYYY-MM-DD
```

Parameters

The `delete_forecast_version_annotations` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the forecast version with the annotations that you want to delete.
`-all`	A parameter that deletes all of the annotations in the forecast version.
`-before` `YYYYMMDD`	A parameter that deletes all of the annotations in the forecast version that were published before the date specified by `YYYY-MM-DD`. For example, specifying `-before 2002-06-04` will delete all annotations that were published before June 6, 2002.

Example

At the command prompt, type:

```
dm::transaction
 {delete_forecast_version_annotations Model Forecast_Version -all}
```

Exceptions

This command throws an exception if no annotations exist in the forecast version or before the specified date.

Copying Forecast Version Notes into Annotations

Forecast version notes are annotations made at a particular time-bucket for a demand point. In the Forecast Studio, you can add notes to a forecast version to inform users about changes that have been made to a demand point during the forecasting process. Annotations, however, apply to the entire demand point. When you publish a scenario that contains notes, the notes are not automatically copied to annotations.

The `copy_forecast_version_notes_into_annotations` command copies the notes in a given forecast version for all demand points in the given hierarchy into the annotations for that forecast version. The command copies one annotation per demand point.

Usage

```
copy_forecast_version_notes_into_annotations model aggregation_hierarchy
forecast_version
```

Parameters

The `copy_forecast_version_notes_into_annotations` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the forecast version with the notes that you want to copy into annotations.
aggregation_hierarchy	The aggregation hierarchy containing the demand points with notes attached to them.
forecast_version	The forecast version containing the notes that you want to copy into the annotations.

Example

At the command prompt, type:

```
dm::transaction
 {copy_forecast_version_notes_into_annotations Bikes top Canada}
```

Exceptions

This command throws an exception if the demand model, aggregation hierarchy, or forecast version does not exist. This command also throws an exception if you do not have write permissions for the demand point into which a note is copied.

Exporting Multiple Forecast Versions

The `export_multiple_forecast_versions` command exports multiple forecast version data from the database into an XML file. Exporting multiple forecast versions is useful to demand planners who want to work with specific forecast data outside the Forecast Studio.

Forecast data that is exported to an XML file can be used in the ForecastWeb Microsoft Excel add-in, which has the same security considerations as the CCR. The Excel client is useful when you want to configure forecast data and create reports outside the application environment.

When setting up the demand point set parameters in the command, ensure that the forecast versions and demand point set security permissions are compatible. For example, assume that a demand planner generates forecast data in the Forecast Studio using the Simple smoothing forecasting model. The demand planner can then export this data into an XML file and review it using the Excel client. The demand planner can change forecast data and then import the data back into the Forecast Studio.

Usage

```
export_multiple_forecast_versions model
aggregation_hierarchy{forecast_version_list}
unit_of_measure filename [demand_point_set]
```

Parameters

The `export_multiple_forecast_versions` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the forecast versions for which forecast data is exported.
aggregation_hierarchy	The name of the aggregation hierarchy in the demand model where the forecast versions reside.
forecast_version_list	The list of forecast versions that are exported into the file.

Parameter	Description
unit_of_measure	The unit of measure in which the forecast data for forecast versions is exported. This unit of measure must be defined in the database before it can be exported successfully.
filename	The name of the file into which forecast versions are exported.
demand_point_set	The name of the demand point set to be exported. All demand points for a forecast version are exported from an aggregation hierarchy and demand model. This parameter is optional.

Example

From the command prompt, type:

```
dm::transaction
{export_multiple_forecast_versions aggregation1 {forecast_version1
forecast_version2} Dollars ForecastProvider.xml RegionalSales}
```

Exceptions

This command throws an exception for these conditions:

- No forecast versions are specified in the forecast version list.
- The demand model does not exist.
- The aggregation hierarchy does not exist.
- The unit of measure does not exist.
- Forecast version and demand point permissions do not match.

Importing Multiple Forecast Versions

The `import_multiple_forecast_versions` command imports forecast data into the database from an XML file. Demand planners who configure forecast data outside of the application can import the contents of their XML file into the database. The forecast data in the XML file must be in the same format as the data in the demand model or in the Excel Add-In.

When forecast data is imported into the database, any imported demand points are overridden by the imported demand points, regardless of security settings. If the import XML file does not contain demand points that exist in the database, the demand points in the database are left unchanged.

The multiple forecast versions data is imported into the demand model using the unit of measure that the demand planner used in the Excel client. If the unit of measure included in the imported data is different than the unit of measure in the database, the imported data is converted to match the database.

Usage

```
import_multiple_forecast_versions model aggregation_hierarchy filename
```

Parameters

The `import_multiple_forecast_versions` command takes these parameters:

Parameter	Description
model	The name of the demand model into which the forecast data is imported.
aggregation_hierarchy	The name of the aggregation hierarchy in the demand model where the forecast versions reside.
filename	The name of the file from which the forecast data is imported. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_multiple_forecast_versions Bikes Sales1 ForecastProvider.xml}
```

Exceptions

This command throws an exception if any of the following do not exist:

- The demand model.
- The aggregation hierarchy.
- The XML file.

Importing Full Forecast Versions

The `import_full_forecast_version` command imports an XML file that was produced by the `export_full_forecast_version` command. The XML file contains forecast data, forecast history, annotations, and reviews that are stored in a forecast version for all associated aggregation hierarchies and units of measure.

Usage

```
import_full_forecast_version model forecast_version filename
```

Parameters

The `import_full_forecast_version` command takes these parameters:

Parameter	Description
model	The name of the demand model into which forecast data is imported.
forecast_version	The name of the forecast version that is imported into the database.
filename	The name of the file from which the forecast version is imported. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
```

```
 {import_full_forecast_version Bikes forecast_version1
ForecastProvider.xml}
```

Exceptions

This command throws an exception for these conditions:

- No forecast version is specified.
- The demand model does not exist.
- The filename does not exist.
- The contents of the XML file do not match the demand model.

Exporting Full Forecast Versions

A demand model can contain a forecast version that exists in several different aggregation hierarchies. This forecast version can display forecast data elements in each aggregation hierarchy in different ways. For example, a forecast version might display forecast data in one aggregation hierarchy using the dollar unit of measure, and it might display the data using the each unit of measure in another aggregation hierarchy.

The `export_full_forecast_version` command exports forecast data, forecast history, and annotations and reviews that are stored in a forecast version for all associated aggregation hierarchies and units of measure into a single XML file.

For example, assume that the Bikes demand model includes several aggregation hierarchies whereby the Market forecast version has forecast data. Each hierarchy uses several units of measure to view the forecast data. The forecast data also contains forecast history data, annotations, and reviews. By using the `export_full_forecast_version` command, you can create one XML file save forecast data that spans multiple aggregation hierarchies and includes different units of measures, forecast history data, annotations, and reviews.

As a default value, the command does not write values for the aggregated time series into the XML file. You must use the`aggregations` parameter if you want to export data for the aggregated time series.

Usage

```
export_full_forecast_version model forecast_version filename
[-aggregations]
```

Parameters

The `export_full_forecast_version` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which forecast data is exported.
`forecast_version`	The name of the forecast version that is exported into the file.
`filename`	The name of the file into which the forecast version is exported. When specifying the filename, provide the path where the file resides.
`[-aggregations]`	A parameter indicating that the export command includes aggregated forecast values.

Example

From the command prompt, type:

```
dm:: transaction
{export_full_forecast_version Bikes forecast_version1 RegionalSales}
```

Exceptions

This command throws an exception for these conditions:

- No forecast version is specified.
- The demand model does not exist.

Disaggregating Forecasts

The `disaggregate` command disaggregates a forecast at a specific aggregation level in the hierarchy. The changes that result from using this command can be pushed up or down the hierarchy, resulting in a balanced forecast.

Usage

```
disaggregate model aggregation_hierarchy forecast_version
 unit_of_measure aggregation_level -system_profile |
-top_down | bottom_up [-balance_above]
```

Parameters

The `disaggregate` command takes the following parameters:

Parameters	Description
`model`	The name of the demand model containing the forecast version that you want to disaggregate.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the forecast version that you want to disaggregate.
`forecast_version`	The name of the forecast version that you want to disaggregate.
`unit_of_measure`	The unit of measure with which you want to disaggregate the forecast.
`aggregation_level`	The aggregation level at which you want to disaggregate the forecast.
`-system_profile`	A flag that forces the`disaggregate` command to distribute the forecast data downward, based on a system-defined pattern.
`-top_down`	A flag that disaggregates forecast data from the aggregation level that you specify down to the lowest level of aggregation.
`-bottom_up`	A flag that disaggregates forecast data from the aggregation level that you specify up to the highest level of aggregation.

Parameters	Description
[-append_notes]	An optional parameter that automatically appends notes to forecasts during the disaggregation process. **Note.** This parameter is only available when using the -top_down disaggregation method.
[-balance_above]	An optional parameter that aggregates above the disaggregated demand point(s). If you use the reconciliation process, it is recommended that you not use this parameter.

Example

```
dm::transaction { disaggregate CanadaParts ProductGroup
Marketing dollars "Total for Company" bottom_up -balance_above }
```

Exceptions

This command throws an exception if any of the following do not exist:

- The demand model.
- The aggregation hierarchy.
- The forecast version.
- The aggregation level.

An exception is also thrown if the selected disaggregation flags are not valid.

Balancing Forecasts

The `balance` command propagates data at the lowest aggregation level to the level that you specify. If any demand points between the two levels have been edited and not disaggregated, this command overwrites them. If you use the reconciliation process, it is advised that you use this command carefully. For example, balance only at the level at which forecast changes are made and not higher.

Usage

```
balance model aggregation_hierarchy forecast_version
aggregation_level
```

Parameters

The `balance` command takes the following parameters:

Parameters	Description
model	The name of the demand model containing the data that you want to propagate to the lowest aggregation level.
aggregation_hierarchy	The name of the aggregation hierarchy that contains the forecast version that you want to balance.
forecast_version	The name of the forecast version that you want to balance.
aggregation_level	The aggregation level at which you want to balance the forecast.

Example

```
dm::transaction { balance EuropeBikes ProductGroupC Sales
 "Product by Customer"}
```

Exceptions

This command throws an exception if any of the following do not exist:

- The demand model.
- The aggregation hierarchy.
- The forecast version.
- The aggregation level.

Copying Forecast Versions

The `copy_forecast_version` command copies a forecast version for all hierarchies and all units of measure.

Usage

```
copy_forecast_version model source_forecast_version
target_forecast_version [-history] [-notes] [-annotations]
```

Parameters

The `copy_forecast_version` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which to copy the forecast version.
`source_forecast_version`	The source demand model's forecast version.
`target_forecast_version`	The target demand model's forecast version.
`[-history]`	The demand model's forecast version history.
`[-notes]`	Miscellaneous notes about the demand model's forecast version.
`[-annotations]`	Explanatory notes about the demand model's forecast version.

Example

At the command prompt, type:

```
dm::transaction
{copy_forecast_version Bikes Sales Marketing}
```

Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Copying Partial Forecast Versions

The `copy_partial_forecast_version` command copies a subset of data from one forecast version to another. It is recommended that the `balance` and/or `disaggregate` commands be used after the `copy_partial_forecast_version` command.

Usage

```
copy_partial_forecast_version model source_forecast_version
target_forecast_version source_bucket_index target_bucket_index
number_of_buckets [aggregation_hierarchy] [demand_point_set]
```

Parameters

The `copy_partial_forecast_version` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which to copy the forecast version.
`source_forecast_version`	The demand model's source forecast version.
`target_forecast_version`	The demand model's target forecast version. **Note.** The target forecast version can be the same as the source.
`source_bucket_index`	An index of the source time bucket that you want to copy to a forecast version, in relation to the current horizon. For example, to start your copy at the tenth period before the forecast, use "-10" as the index.
`target_bucket_index`	An index of the target time bucket, in relation to the current horizon. For example, to copy the first period of the forecast, use "0" as the index.
`number_of_buckets`	The number of time buckets that you want to copy from the source to the target forecast.
`aggregation_hierarchy`	The name of the aggregation hierarchy to copy.
`demand_point_set`	The name of the demand point set to copy. This parameter is option, if no demand point set is specified, all demand points are copied.

Example

At the command prompt, type:

```
dm::transaction
{copy_partial_forecast_version Bikes Sales Reconciled 5 17 1}
```

This command copies the sixth forecast period to the eighteenth forecast period.

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The source or destination forecast version does not exist.
- The demand point set does not exist in the demand model.

Renaming Forecast Versions

The `rename_forecast_version` command renames a forecast version in an existing demand model.

Usage

```
rename_forecast_version model source-forecast-version target-forecast-version
```

Parameters

The `rename_forecast_version` command takes these parameters:

Parameter	Description
`model`	The name of the demand model in which the forecast version is renamed.
`source-forecast-version`	The source demand model's forecast version.
`target-forecast-version`	The target demand model's forecast version.

Example

At the command prompt, type:

```
dm::transaction
{rename_forecast_version Bikes Consensus Sales}
```

Exceptions

This command throws an exception if the demand model or the forecast version does not exist.

Summing Forecast Versions

The `sum_forecast_versions` command enables you to model multiple forecast versions that are incremental. You can use this command to sum data for multiple forecast versions and publish the data to another forecast version.

Usage

```
sum_forecast_versions model target_forecast_version
[-hierarchy <hierarchy_name>] [-source <forecast_version1>]
[-source <forecast_version2>] ...[-source <forecast_versionN>]
```

Parameters

The `sum_forecast_versions` command takes these parameters:

Parameter	Description
`model`	The name of the demand model for which you want to sum forecast versions.
`target_forecast_version`	The name of the forecast version that contains the sum of the source forecast versions.

Parameter	Description
[-hierarchy <hierarchy_name>]	The name of the aggregation hierarchy.
[-source <forecast_versionN>]	The name of a source forecast version to include in the sum of the source forecast versions. Two or more forecast versions are supported.

Example

At the command prompt, type:

```
sum_forecast_versions
Bikes "Total Forecast" "Baseline Forecast"
"Lift Forecast"

where the sum of the forecasts contained in the
Baseline Forecast and the Lift Forecast forecast
versions are copied to the
Total Forecast forecast version.
```

Exceptions

This command throws an exception if the demand model or forecast versions do not exist.

Generating Forecast Change Reports

The `report_forecast_change` command generates a forecast change report by querying forecast change details saved as annotations (comments).

Usage

```
report_forecast_change model hierarchy forecast_version
unit_of_measure filename [-xml] [-demand_point_set <dpsName>]
[-absolute_difference <value>] [-absolute_percent_difference <value>]
[-timestamp_days <numberOfDays>] [-onlyLastChange]
[-user <userName] [-time_periods
<startPeriod> <endPeriod>]
```

Parameters

The `report_forecast_change` command takes these parameters:

Parameter	Description
model	The name of the demand model for which you want to generate a forecast change report.
hierarchy	The name of the aggregation hierarchy.
forecast_version	The name of the forecast version on which you want to base the report.
unit_of_measure	The unit of measure on which you want to base the report.
filename	The name of the XML or CSV file containing the generated report. When specifying the filename, provide the path where the file resides.

Parameter	Description
[-xml]	Specifies that the output be written in xml format. This parameter is optional and output is written in CSV format by default.
[-demand_point_set <dpsName>]	The name of the demand point set by which to restrict the report. This parameter is optional and if no demand point set is specified, then all demand points are considered.
[-absolute_difference <value>]	The forecast change quantity threshold. If the absolute difference between the new forecast quantity and the old forecast quantity is equal to or greater than the absolute_difference threshold value, then the data point is included in the forecast change report. This parameter is optional.
[-absolute_percent_difference <value>]	The forecast change percentage threshold. If the absolute percentage difference between the new forecast quantity and the old forecast quantity (where the difference is divided by the old forecast quantity) is equal to or greater than the absolute_percent_difference threshold value, then the data point is included in the forecast change report. This parameter is optional.
[-timestamp_days <numberOfDays>]	The number of days prior to the report run date and time to report on forecast changes. For example, if the report command is executed on 2008-07-15 22:00:00 and timestamp_days is 7, then the forecast change data points with timestamps within the last 7 days of 2008-07-15 22:00:00 are included in the forecast change report. This parameter is optional.
[-onlyLastChange]	If more than one forecast change was made to the same data point for the report range, then only the last change is considered. This parameter is optional.
[-user <userName]	The name of the user. You can restrict the report to forecast changes made by a specific user. This parameter is optional. If no user is specified, then all user names are considered.
[-time_periods <startPeriod> <endPeriod>]	The period range in relation to the current horizon start period. You can restrict the report to forecast changes made in a specific period range. For example, the first period bucket of the forecast range has an index value of "0" and the most recent period of forecast history has an index value of "-1". To generate a report that considers only the last three periods of the forecast history and the first three periods of forecast, then specify startPeriod equal to "-3" and endPeriod equal to 2. This parameter is optional. If no period range is specified, then all periods are considered.

Example

From the command prompt, type:

```
% dm::transaction
{report_forecast_change Bikes Default Sales Each
d:/dm/reports/ForecastChange_Bikes_Brian.csv
-absolute_difference 50 -absolute_percent_difference 20
-timestamp_days 3 -onlyLastChange -user brian -time_periods 2 5}
```

```
where absolute_difference is 50, absolute_percent_difference is 20,
timestamp_days is 3 and the period range starts with the third
forecast period and ends with the sixth forecast period.
```

Exceptions

This command throws an exception if the demand model does not exist.

Effective Date Management

This section provides an overview of effective date management commands and discusses how to:

- Export effective dates.
- Import effective dates.
- Reset all effective dates.

Understanding Effective Date Management Commands

You can use these commands to manage effective date information:

- `export_effective_dates`
- `import_effective_dates`
- `reset_all_effectivity_dates`

Exporting Effective Dates

The `export_effective_dates` command exports effective dates from the demand model for each demand point that effective start or effective end dates use into an XML file.

Usage

```
export_effective_dates model filename
```

Parameters

The `export_effective_dates` command takes these parameters:

Parameters	Description
`model`	The name of the demand model where you want to import the business model.
`filename`	The name of the file to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{export_effective_dates Sales BikesModel.xml}
```

Exceptions

This command throws an exception if the model does not exist.

Importing Effective Dates

The `import_effective_dates` command imports effective dates into a demand model from an XML file or demand points listed in an XML file. Demand points that are not listed in the file are left intact.

Usage

```
import_effective_dates model filename
```

Parameters

The `import_effective_dates` command takes these parameters:

Parameter	Description
`model`	The name of the demand model where you want to import the demand model.
`filename`	The name of the XML file to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_effective_dates Sales BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The import fails.
- The effective date range is not valid.
- The demand point does not exist in the demand model.

Resetting All Effective Dates

The `reset_all_effective_dates` command to reset the effective start and effective end dates for all demand points in the demand model.

Usage

```
reset_all_effective_dates model
```

Parameters

The `reset_all_effective_dates` command takes this parameter:

Parameter	Description
model	The name of the demand model where you want to import the business model.

Example

From command prompt, type:

```
dm::transaction
 {reset_all_effective_dates Sales}
```

Scenario Data Management

This section provides an overview of scenario data management commands and discusses how to:

- Export scenarios.
- Import scenarios.
- List scenarios.
- Delete scenarios.
- Run scenarios.
- Publish scenarios.
- Add demand points to scenarios.
- Repair outliers.

Understanding Scenario Data Management Commands

You can use these commands to manage scenario data:

- export_scenarios
- import_scenarios
- list_scenarios
- delete_scenario
- run_scenario
- publish_scenario

Exporting Scenarios

The export_scenario command extracts the information from a scenario and writes it to an XML file with a name that you specify.

Usage

```
export_scenario model aggregation_hierarchy scenario filename
```

Parameters

The `export_scenario` command takes these parameters:

Parameter	Description
`model`	The name of the demand model that contains the scenario to export.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the scenario.
`scenario`	The name of the scenario to export.
`filename`	The name of the XML file into which this command writes the information in the scenario. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_scenario Model Sales EasternSales BikesModel.xml}
```

Exceptions

This command throws an exception if the demand model or the scenario do not exist.

Importing Scenarios

The `import_scenario` command imports a scenario from an XML file (either on a local disk or a network drive). If the scenario already exists in the model, this command overwrites the information from the existing scenario with the information in the XML file. If no scenario exists with the specified name, the `import_scenario` command creates a new scenario with the name that you specify.

Usage

```
import_scenario model aggregation_hierarchy scenario filename
```

Parameters

The `import_scenario` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the scenario.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the scenario.
`scenario`	The name of the scenario to import.
`filename`	The name of the XML file containing the scenario to import. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_scenario Model Sales EasternSales BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The file format is invalid.

Listing Scenarios

The `list_scenarios` command gets or sets a list of the scenarios that a particular demand model contains. This command is useful when you want to export or delete specific scenarios from a demand model.

The `list_scenarios` command returns a list in a format similar to this example:

```
{Optimistic Pessimistic {Italy 1} {Powertrains 1} {Powertrains 2} {Powertrains 3}}
```

If a scenario name contains spaces, Tcl encloses the name in curly braces.

Usage

```
list_scenarios model aggregation_hierarchy
```

Parameters

The `list_scenarios` command takes this parameter:

Parameter	Description
`model`	The name of the demand model containing the scenarios that you want to list.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the scenario.

Example

From the command prompt, type:

```
dm::transaction
 {list_scenarios Model EasternSales}
```

Exceptions

This command throws an exception if the model does not exist.

Deleting Scenarios

The `delete_scenario` command deletes scenarios from a demand model.

Usage

```
delete_scenario model aggregation_hierarchy scenario
```

Parameters

The `delete_scenario` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the scenario that you want to delete.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the scenario.
`scenario`	The name of the scenario to delete.

Example

From the command prompt, type:

```
dm::transaction
 {delete_scenario ConferenceRoom Sales EasternSales}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The demand model does not contain the scenario.

Running Scenarios

The `run_scenario` command generates a forecast for all scenario points for the given scenario. If the history series contains just missing values for a particular scenario point, a warning message appears containing the full product, location, and channel path for that scenario point. If another low-level error occur, this command throws an exception and the changes in the transaction are rolled back if the errors are not corrected.

Usage

```
run_scenario model aggregation_hierarchy scenario
```

Parameters

The `run_scenario` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the scenario for which the forecast is generated.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the scenario.
`scenario`	The name of the scenario on which the`run_scenario` command generates a forecast.

Example

From the command prompt, type:

```
dm::transaction {run_scenario model Sales1 EasternSales}
```


Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist.
- The scenario does not exist.

Publishing Scenarios

The `publish_scenario` command publishes scenarios to a forecast version. This command copies all adjusted forecasts from the scenario into the forecast version. Doing this makes the forecast visible to other users of the application.

Usage

```
publish_scenario model aggregation_hierarchy scenario forecast_version
```

Parameters

The `publish_scenario` command uses these parameters:

Parameter	Description
`model`	The name of the demand model containing the scenario to publish.
`aggregation_hierarchy`	The name of the aggregation hierarchy that contains the scenario.
`scenario`	The name of the scenario.
`forecast_version`	The name of the forecast version into which the adjusted forecast will be copied.

Example

From the command prompt type:

```
dm::transaction {publish_scenario RegionalSales Optimistic Optimistic_Forecast}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The scenario does not exist.
- The forecast version does not exist.

Adding Demand Point Sets to Scenarios

The `add_set_to_scenario` command adds an existing demand point set to the specified scenario. This command also creates a scenario with the demand points from the demand point set if it does not already exist.

Adding demand points to scenarios involves specifying the scenario and model name to which you want to import the set. You must create the specified demand point sets before importing.

You can add multiple demand point sets to a scenario; use the `add_set_to_scenario` command for each set that you want to add.

Usage

```
add_set_to_scenario model aggregation_hierarcy scenario demand_point_set
unit_of_measure [-use_base]
```

Parameters

The `add_set_to_scenario` command takes these parameters:

Parameter	Description
`model`	The name of the demand model.
`aggregation_hierarchy`	The name of the aggregation hierarchy containing the scenario.
`scenario`	The name of the scenario to which you want to import demand points.
`demand_point_set`	The name of the demand point set that you want to add to the scenario.
`unit_of_measure`	The unit of measure in which the demand point set is defined.
`-use_base`	An optional flag that allows you to use the base unit of measure for leaf demand points.

Example

From the command prompt type:

```
dm::transaction
{CanadaSales Regional TotalSales Marketing Each -use_base}
```

This command throws an exception if the following do not exist:

- Demand model.
- Aggregation hierarchy.
- Scenario.
- Demand point set.
- Unit of measure.

Repairing Outliers

The `repair_outliers` command mitigates the effects of outliers on a forecast. Repairing outliers involves specifying whether you want to apply the fixes to the entire scenario publishes scenarios to a forecast version. You can also specify the number of standard deviations to which the outliers are moved which controls the amount by which the effects of the outliers are mitigated.

Usage

```
repair_outliers model aggregation_hierarchy scenario
new_standard_deviation [-reforecast]
```

Parameters

The `repair_outliers` command takes these parameters:

Parameter	Description
`model`	The name of the demand model.
`aggregation_hierarchy`	The name of the aggregation hierarchy containing the scenario with outliers.
`scenario`	The name of the scenario that contains the outliers that you want to repair.
`new_standard_deviation`	The number of standard deviations from the historical fit that the outlier will be moved. If you type zero, the outliers will be aligned with the historical fit.
`-reforecast`	An optional flag that enables you to generate a new forecast after the outliers are repaired.

Example

From the command prompt type:

```
dm::transaction
 {repair_outliers CanadaSales Regional TotalSales 4 -reforecast}
```

Exceptions

This command throws an exception if the following do not exist:

- Demand model.
- Aggregation hierarchy.
- Scenario.

Batch Queue Management

This section provides an overview of batch queue management commands and discusses how to:

- List jobs.
- Delete a job.
- Delete submitted jobs.
- Stop the batch server.

Understanding Batch Queue Management Commands

You can use these commands to control the batch queue:

- `list_jobs`
- `delete_ job`
- `delete_jobs`

- `stop_batch_server`

Listing Jobs

The `list_jobs` command returns a list with the names of report jobs for a demand model held in the batch queue.

Usage

```
list_jobs model
```

Parameters

The `list_jobs` command takes this parameter:

Parameter	Description
`model`	The name of the demand model for which you want to list the report jobs.

Example

From the command prompt, type:

```
dm::transaction {list_jobs ConferenceRoom}
```

Exceptions

This command throws an exception if the demand model does not exist.

Deleting a Job

The `delete_job` command deletes a job for a demand model from the batch queue.

Usage

```
delete_job model id
```

Parameters

The `delete_job` command takes these parameters:

Parameters	Description
`model`	The name of the demand model for which you want to delete the report jobs.
`id`	The ID of the job that you want to delete.

Example

From the command prompt, type:

```
dm::transaction
{delete_job ConferenceRoom Job1}
```

Exceptions

This command throws an exception if the demand model does not exist.

Deleting Submitted Jobs

The `delete_jobs` command deletes a set of submitted jobs from the batch queue. Using flags, you can specify which jobs to delete from the queue.

Usage

```
delete_jobs model -all -pending -executing -failed
```

Parameters

The `delete_jobs` command takes these parameters:

Parameter	Description
`model`	The name of the demand model.
`-all`	Deletes all of the jobs in the batch queue.
`-pending`	Deletes all pending jobs in the batch queue.
`-executing`	Deletes all jobs in the batch queue that are being executed.
`-failed`	Deletes all failed jobs in the batch queue.

Example

From the command prompt, type:

```
dm::transaction
 {delete_jobs ConferenceRoom -all}
```

Exceptions

This command throws an exception if the demand model does not exist.

Stopping the Batch Server

The `stop_batch_servers` command stops the batch server from running and processing jobs in the batch queue. Stopping the batch server does not clear submitted jobs from the batch queue.

Usage

```
stop_batch_servers model
```

Parameters

The `stop_batch_servers` command takes this parameter:

Reference	Description
`model`	The name of the demand model for which the batch server will stop running.

Example

From the command prompt, type:

```
dm::transaction
 {stop_batch_servers model}
```

Reconciliation Management

This section provides an overview of reconciliation management commands and discusses how to:

- Calculate reconciliation weights.
- Continue calculating reconciliation weights.
- Import reconciliation weights.
- Import a weighting category.
- Export a weighting category.
- Import a forecast weighting category.
- Export a forecast weighting category.

Understanding Reconciliation Management Commands

Reconciliation is the process of comparing and combining multiple forecasts to create one enterprise forecast. Forecast data often contains conflicting forecast data values. Distinguishing between accurate and inaccurate forecast data is critical to the collaborative process. The reconciliation process provides an opportunity for demand planners or other system users to use multiple forecasts to create an accurate enterprise forecast that is based on input from several sources. Use the reconciliation management Demand Automation Shell commands to manage reconciliation weights and forecast weighting categories.

Before you can use any of the reconciliation management commands you need to set up a demand model with forecast versions that contain forecast data originally created by the different users and actual sales numbers.

You can use these commands to manage reconciliations:

- `calculate_weights`
- `continue_calculate_weights`
- `import_reconcilation_weights`
- `import_weighting_category`
- `export_weighting_category`
- `import_forecast_weighting_category`
- `export_forecast_weighting_category`

Calculating Reconciliation Weights

The `calculate_weights` command calculates reconciliation weights for forecast versions in the demand model and stores the result into the `result_filename` output file. The command runs an algorithm that calculates a weighted least squares regression value for each forecast version. The algorithm is iterative and continuously improves the weights at each step. Therefore, you can stop the calculation after a fixed number of seconds, and then continue the calculation to further improve the solution.

Note. When you run the `calculate_weights` command, any reconciliation weights that users have defined are overridden.

Usage

```
calculate_weights model result_filename
unit_of_measure max_duration start_period end_period
forecast_version1 forecast_version2...
```

Parameters

The `calculate_weights` command takes these parameters:

Parameter	Description
`model`	The name of the demand model for which you want to calculate reconciliation weights.
`results_filename`	The name of a file to write calculation results to. This file can be used to continue the calculation at a later time.
`unit_of_measure`	The unit of measure for the forecasts to use in the calculation.
`max_duration`	The maximum number of seconds to run the calculation.
`start_period`	The index of the first time bucket to use in the calculation, starting at 0.
`end_period`	The index of the last time bucket to use in the calculation, starting at 0.
`forecast_version1`	The name of a forecast version for which weights are calculated.

Example

From the command prompt, type:

```
dm::transaction
 {calculate_weights Model results.xml dollars 480 1 4 marketing}
```

Exceptions

The command throws an exception for these conditions:

- The demand model does not exist.
- The forecast versions (`forecast_version1 forecast_version2`?) do not exist.
- An invalid unit of measure was specified.
- The pair `start_period` and `end_period` is not valid (1<= `start_period < end_period`) and does not belong in the sales history horizon.

Continue Calculating Reconciliation Weights

The `continue_calculate_weights` command continues a weighting calculation from the point at which the calculation stopped in a previous run.

Usage

```
continue_calculate_weights model input_results_filename
output_results_filename max_duration
```

Parameters

The `continue_calculate_weights` command takes these parameters:

Parameter	Description
`model`	The name of the demand model for which you want to continue calculating reconciliation weights.
`input_results_filename`	The file name of a previously run instance of the`calculate_weights` command or the`continue_calculate_weights` command.
`output_results_filename`	The file name to which the calculation results are written. This file can be used to continue the calculation at a later time.
`max_duration`	The maximum number of seconds to run the calculation.

Example

From the command prompt, type:

```
dm::transaction
 {continue_calculate_weights Model results.xml outcome.xml 480}
```

Exceptions

The command throws an exception for these conditions:

- The demand model does not exist.
- The forecast versions (`forecast_version1 forecast_version2`?) do not exist.
- An invalid unit of measure was specified.
- The pair `start_period` and `end_period` is not valid (1<= `start_period < end_period`) and does not belong in the sales history horizon.

Import Reconciliation Weights

The `import_reconcile_weights` command imports reconciliation weights into a model from an XML file.

Usage

```
import_reconcile_weight model filename
```

Parameters

The `import_reconcile_weights` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import reconciliation weights.
filename	The name of the XML file from which to get or set the reconciliation weights. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_reconcile_weights Model import.xml}
```

Importing a Weighting Category

The `import_weighting_category` command imports a weighting category into a demand model. The import file contains a list of names of the weighting categories and the relative weight that each weighting category is assigned.

Usage

```
import_weighting_category model filename
```

Parameters

The `import_weighting_category` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import a weighting category.
filename	The name of the file into which to get or set the weighting category. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_weighting_category Model import.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Exporting a Weighting Category

The `export_weighting_category` command exports a weighting category into a demand model. The export file contains a list of the names of the weighting categories and the relative weight that each weighting category is assigned.

Usage

```
export_weighting_category model filename
```

Parameters

The `export_weighting_category` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export a weighting category.
`filename`	The name of the file into which the system exports the weighting category. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {export_weighting_category Model export.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing a Forecast Weighting Category

The `import_forecast_weighting_category` command imports a forecast weighting category into a demand model from a file. The import file contains a list of demand points and the category to which each demand point is associated. Existing weighting categories are overwritten when new weighting categories are imported into the demand model.

Usage

```
import_forecast_weighting_category model filename
```

Parameters

The `import_forecast_weighting_category` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import a forecast weighting category.
`forecast_version`	The name of the forecast version.
`filename`	The name of the file into which to import the forecast weighting category. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_forecast_weighting_category Model Sales import.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The forecast version does not exist in the model.

Exporting a Forecast Weighting Category

The `export_forecast_weighting_category` command exports a forecast weighting category from a demand model into a file. On export, the file contains a list of demand points and the category to which each demand point is associated.

Usage

```
export_forecast_weighting_category model filename
```

Parameters

The `export_forecast_weighting_category` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export a forecast weighting category.
`forecast_version`	The name of the forecast version.
`filename`	The name of the file into which to export a weighting category. When specifying the filename, provide the path where the file resides.

Examples

From the command prompt, type:

```
dm::transaction
 {export_forecast_weighting_category Model Sales export.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Predictor Management

This section provides an overview of predictor management commands and discusses how to:

- Import predictors.
- Export predictors.
- Import the assignment of predictors to demand points in a specific hierarchy.
- Export the assignment of predictors to demand points in a specific hierarchy.
- Import predictors assigned to all demand points.

- Export predictors assigned to all demand points.

Understanding Predictor Management Commands

There are a variety of Demand Automation Shell commands that enable you to manage the data associated with predictors. These commands are grouped as follows; importing and exporting sets of predictors that exist in a demand model, importing and exporting the assignment of predictors to specific demand points in a model for a given hierarchy, and importing and exporting the assignment of predictors to all demand points in a model for all hierarchies.

You can use these commands to manage predictors:

- `import_predictors`
- `export_predictors`
- `import_demand_point_predictors_assignment`
- `export_demand_point_predictors_assignment`
- `import_all_demand_point_predictors_assignment`
- `export_all_demand_point_predictors_assignment`

Importing Predictors

The `import_predictors` command imports a set of predictors into a demand model from an XML file (either on a local drive or a network drive). If a predictor exists in the model, then this command overwrites the existing information with information from the XML file. If a predictor does not exist in the model, then this command creates it in the model.

Usage

```
import_predictors model filename
```

Parameters

The `import_predictors` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which this command imports the predictors.
`filename`	The name of the XML file containing the predictors. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_predictors ConferenceRoom predictors.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.

- The XML file does not exist.

Exporting Predictors

The `export_predictors` command exports a set of predictors from a demand model to an XML file (either on a local drive or a network drive).

Usage

```
export_predictors model filename
```

Parameters

The `export_predictors` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the predictors that you want to export.
`filename`	The name of the XML file into which you want to export the predictors. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
 {export_predictors ConferenceRoom predictors.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Predictor Assignment to Demand Points in a Specific Hierarchy

The `import_demand_point_predictors_assignment` command imports predictors that are assigned to demand points. Predictors can be assigned to demand points within a demand model and within a specific aggregation hierarchy. The command imports the predictor assignment information from an XML file (either on a local drive or a network drive).

On import, any predictors that are assigned to specific demand points will be assigned to the corresponding scenario points in the demand model.

Usage

```
import_demand_point_predictors_assignment model hierarchy filename
```

Parameters

The `import_demand_point_predictors_assignment` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import the predictor assignment information.
hierarchy	The name of the aggregation hierarchy in the demand model to which the assignment information will be added.
filename	The name of the XML file containing the predictor assignment information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_demand_point_predictors_assignment
ConferenceRoom predictors.xml}
```

Exceptions

This command throws an exception if any of the following do not exist:

- Demand model.
- Hierarchy.
- XML file.

Exporting Predictor Assignments to Demand Points in a Specific Hierarchy

The `export_demand_point_predictors_assignment` command exports predictor assignment information to an XML file (either on a local drive or a network drive).

Usage

```
export_demand_point_predictors_assignment model hierarchy filename
```

Parameters

The `export_demand_point_predictors_assignment` command takes these parameters:

Parameter	Description
model	The name of the demand model from which you want to export the predictor assignment information.
hierarchy	The name of the aggregation hierarchy in the demand model containing the assignment information that you want to export.
filename	The name of the XML file to which you want to export the predictor assignment information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_demand_point_predictors_assignment ConferenceRoom WinterPromo⇒
 predictors.xml}
```

Exceptions

This command throws an exception is any of the demand model or the hierarchy do not exist.

Importing Predictors Assigned to All Demand Points

The `import_all_demand_point_predictors_assignment` command imports predictors that are assigned to demand points. The command imports the predictor assignment information from an XML file (either on a local drive or a network drive).

Note. This command is used primarily when restoring a demand model. The difference between this command and the `import_all_demand_point_predictors_assignment` command is that this command works faster because it processes all hierarchies at once.

Usage

```
import_all_demand_point_predictors_assignment model filename
```

Parameters

The `import_all_demand_point_predictors_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the predictor assignment information.
`filename`	The name of the XML file containing the predictor assignment information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction {
import_all_demand_point_predictors_assignment ConferenceRoom predictors.xml
}
```

Exceptions

This command throws an exception if any of the demand models or the XML file do not exist.

Exporting Predictors Assigned to All Demand Points

The `export_all_demand_point_predictors_assignment` command exports predictors that are assigned to demand points to an XML file (either on a local drive or a network drive).

Note. This command is used primarily when backing up a demand model. The difference between this command and the `export_all_demand_point_predictors_assignment` command is that this command works faster because it processes all hierarchies at once.

Usage

```
export_all_demand_point_predictors_assignment model filename
```

Parameters

The `export_all_demand_point_predictors_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the predictor assignment information.
`filename`	The name of the XML file into which you want to export the predictor assignment information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
 {export_all_demand_point_predictors_assignment ConferenceRoom predictors.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Event Management

This section provides an overview of event management commands and discusses how to:

- Import events.
- Export events.
- Import the assignment of events to demand points in a specific hierarchy.
- Export the assignment of events to demand points in a specific hierarchy.
- Import events assigned to all demand points.
- Export events assigned to all demand points.

Understanding Event Management Commands

There are a variety of Demand Automation Shell commands that enable you to manage the data associated with events. These commands are grouped as follows; importing and exporting sets of events that exist in a demand model, importing and exporting the assignment of events to specific demand points in a model for a given hierarchy, and importing and exporting the assignment of events to all demand points in a model for all hierarchies.

You can use these commands to manage events:

- `import_events`
- `export_events`

- import_demand_point_events_assignment
- export_demand_point_events_assignment
- import_all_demand_point_events_assignment
- export_all_demand_point_events_assignment

Understanding Event Management Commands

There are a variety of Demand Automation Shell commands that enable you to manage the data associated with events. These commands are grouped as follows; importing and exporting sets of events that exist in a demand model, importing and exporting the assignment of events to specific demand points in a model for a given hierarchy, and importing and exporting the assignment of events to all demand points in a model for all hierarchies.

You can use these commands to manage events:

- import_events
- export_events
- import_demand_point_events_assignment
- export_demand_point_events_assignment
- import_all_demand_point_events_assignment
- export_all_demand_point_events_assignment

Importing Events

The import_events command imports a set of events into a demand model from an XML file (either on a local drive or a network drive). If an event exists in the model, then this command overwrites the existing information with information from the XML file. If an event does not exist in the model, then this command creates it in the model.

Usage

```
import_events model filename
```

Parameters

The import_events command takes these parameters:

Parameter	Description
model	The name of the demand model into which this command imports the events.
filename	The name of the XML file containing the events. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_events ConferenceRoom events.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The XML file does not exist.

Exporting Events

The `export_events` command exports a set of events from a demand model to an XML file (either on a local drive or a network drive).

Usage

```
export_events model filename
```

Parameters

The `export_events` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the events that you want to export.
`filename`	The name of the XML file into which you want to export the events. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_events ConferenceRoom events.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Event Assignments of Demand Points in a Specific Hierarchy

The `import_demand_point_events_assignment` command imports event assignment information from an XML file (either on a local drive or a network drive). On import, any events that are assigned to specific demand points will be assigned to the corresponding scenario points in the demand model.

Usage

```
import_demand_point_events_assignment model hierarchy filename
```

Parameters

The `import_demand_point_events_assignment` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import the event assignment information.
hierarchy	The name of the aggregation hierarchy in the demand model to which the assignment information will be added.
filename	The name of the XML file containing the event assignment information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_demand_point_events_assignment ConferenceRoom WinterPromo  events.xml}
```

Exceptions

This command throws an exception if any of the following do not exist:

- Demand model.
- Event.
- Hierarchy.
- XML file.

Exporting the Assignment of Events to Demand Points in a Specific Hierarchy

The `export_demand_point_events_assignment` command exports event assignment information to an XML file (either on a local drive or a network drive).

Usage

```
export_demand_point_events_assignment model hierarchy filename
```

Parameters

The `export_demand_point_events_assignment` command takes these parameters:

Parameter	Description
model	The name of the demand model from which you want to export the event assignment information.
hierarchy	The name of the aggregation hierarchy in the demand model containing the assignment information that you want to export.
filename	The name of the XML file to which you want to export the event assignment information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
 {export_demand_point_events_assignment ConferenceRoom WinterPromo events.xml}
```

Exceptions

This command throws an exception if the any of the demand model or the hierarchy do not exist.

Importing Events Assigned to All Demand Points

The `import_all_demand_point_events_assignment` command imports event assignment information from an XML file (either on a local drive or a network drive).

Note. This command is used primarily when restoring a demand model. The difference between this command and the `import_all_demand_point_events_assignment` command is that this command works faster because it processes all hierarchies at once.

Usage

```
import_all_demand_point_events_assignment model filename
```

Parameters

The `import_all_demand_point_events_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the event assignment information.
`filename`	The name of the XML file containing the event assignment information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_all_demand_point_events_assignment ConferenceRoom events.xml}
```

Exceptions

This command throws an exception if any of the demand model or the XML file do not exist.

Exporting Events Assigned to All Demand Points

The `export_all_demand_point_events_assignment` command exports event assignment information to an XML file (either on a local drive or a network drive).

Note. This command is used primarily when backing up a demand model. The difference between this command and the `export_all_demand_point_events_assignment` command is that this command works faster because it processes all hierarchies at once.

Usage

```
export_all_demand_point_events_assignment model filename
```

Parameters

The `export_all_demand_point_events_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the event assignment information.
`filename`	The name of the XML file into which you want to export the event assignment information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_all_demand_point_events_assignment ConferenceRoom events.xml}
```

Intervention Management

This section provides an overview of intervention management commands and discusses how to:

- Import interventions.
- Export interventions.
- Import the assignment of interventions to demand points in a specific hierarchy.
- Export the assignment of interventions to demand points in a specific hierarchy.
- Import interventions assigned to all demand points.
- Export interventions assigned to all demand points.

Understanding Intervention Management Commands

There are a variety of Demand Automation Shell commands that enable you to manage the data associated with interventions. These commands are grouped as follows; importing and exporting sets of interventions that exist in a demand model, importing and exporting the assignment of interventions to specific demand points in a model for a given hierarchy, and importing and exporting the assignment of interventions to all demand points in a model for all hierarchies.

You can use these commands to manage interventions:

- `import_interventions`
- `export_interventions`
- `import_demand_point_interventions_assignment`
- `export_demand_point_interventions_assignment`
- `import_all_demand_point_interventions_assignment`
- `export_all_demand_point_interventions_assignment`

Importing Interventions

The `import_interventions` command imports a set of interventions from an XML file (either on a local drive or a network drive). If an intervention exists in the model, then this command overwrites the existing information with information from the XML file. If an intervention does not exist in the model, then this command creates it in the model.

Usage

```
import_interventions model filename
```

Parameters

The `import_interventions` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which this command imports the interventions.
`filename`	The name of the XML file containing the interventions. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_interventions ConferenceRoom interventions.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The XML file does not exist.

Exporting Interventions

The `export_interventions` command exports a set of interventions to an XML file (either on a local drive or a network drive).

Usage

```
export_interventions model filename
```

Parameters

The `export_interventions` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the interventions that you want to export.
filename	The name of the XML file into which you want to export the interventions. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
 {export_interventions ConferenceRoom interventions.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Importing Intervention Assignments to Demand Points in a Specific Hierarchy

The `import_demand_point_interventions_assignment` command imports intervention assignment information from an XML file (either on a local drive or a network drive). On import, any interventions that are assigned to specific demand points will be assigned to the corresponding scenario points in the demand model.

Usage

```
import_demand_point_interventions_assignment model hierarchy filename
```

Parameters

The `import_demand_point_interventions_assignment` command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import the intervention assignment information.
hierarchy	The name of the aggregation hierarchy in the demand model to which the assignment information will be added.
filename	The name of the XML file containing the intervention assignment information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_demand_point_interventions_assignment ConferenceRoom WinterPromo
 interventions.xml}
```

Exceptions

This command throws an exception if any of the following do not exist:

- Demand model.
- Intervention.
- Hierarchy.
- XML file.

Exporting the Intervention Assignments to Demand Points in a Specific Hierarchy

The `export_demand_point_interventions_assignment` command exports intervention assignment information to an XML file (either on a local drive or a network drive).

Usage

```
export_demand_point_interventions_assignment model hierarchy filename
```

Parameters

The `export_demand_point_interventions_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the intervention assignment information.
`hierarchy`	The name of the aggregation hierarchy in the demand model containing the assignment information that you want to export.
`filename`	The name of the XML file to which you want to export the intervention assignment information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction {export_demand_point_interventions_assignment ConferenceRoom⇒
 interventions.xml}
```

Exceptions

This command throws an exception if any of the demand model or the hierarchy do not exist.

Importing Interventions Assigned to All Demand Points

The `import_all_demand_point_interventions_assignment` command imports the intervention assignment information from an XML file (either on a local drive or a network drive).

Note. This command is used primarily when restoring a demand model. The difference between this command and the `import_all_demand_point_interventions_assignment` command is that this command works faster because it processes all hierarchies at once.

Usage

```
import_all_demand_point_interventions_assignment model filename
```

Parameters

The `import_all_demand_point_interventions_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the intervention assignment information.
`filename`	The name of the XML file containing the intervention assignment information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_all_demand_point_interventions_assignment ConferenceRoom interventions.xml}
```

Exceptions

This command throws an exception if any of the demand model or the XML file do not exist.

Exporting Interventions Assigned to All Demand Points

The `export_all_demand_point_interventions_assignment` command exports intervention assignment information to an XML file (either on a local drive or a network drive).

Note. This command is used primarily when backing up a demand model. The difference between this command and the `export_all_demand_point_interventions_assignment` command is that this command works faster because it processes all hierarchies at once.

Usage

```
export_all_demand_point_interventions_assignment model filename
```

Parameters

The `export_all_demand_point_interventions_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the intervention assignment information.
`filename`	The name of the XML file into which you want to export the intervention assignment information. When specifying the filename, provide the path where the file should reside.

Example

From the Windows command prompt, type:

```
dm::transaction
{export_all_demand_point_interventions_assignment ConferenceRoom interventions.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Sales History Data Management

This section provides an overview of sales history data management commands and discusses how to:

- Clear sales history.
- Export sales history.
- Import sales history.
- Copy sales history to a forecast version.

Understanding Sales History Data Management Commands

You can use these commands to manage sales history data:

- `clear_history`
- `export_sales_history`
- `import_sales_history`
- `copy_sales_to_forecast_version`

Clearing Sales History

The `clear_history` command clears sales forecasts from a demand model.

Usage

```
clear_history model
```

Parameters

The `clear_history` command takes this parameter:

Parameter	Description
`model`	The name of the demand model from which you want to clear the sales history.

Example

From the command prompt, type:

```
dm::transaction
 {clear_history Model}
```

Exceptions

The command throws an exception if the demand model does not exist.

Exporting Sales History

The `export_history` command exports the sales history in a demand model (using a specified unit of measure) into an output file.

Usage

```
export_history model unit_of_measure
filename [-aggregates]
```

Parameters

The `export_history` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the sales history.
`unit_of_measure`	The unit of measure in which the sales history data is defined.
`filename`	The name of the file to which data is exported. When specifying the filename, provide the path where the file should reside.
`[-aggregates]`	Exports aggregated values that are calculated. Without this parameter, the system exports all values.

Example

From the command prompt, type:

```
dm::transaction
{export_history Model dollars saleshistory.xml}
```

Exceptions

The command throws an exception for these conditions:

- The demand model does not exist.
- An invalid unit of measure is specified.
- The export into the file fails.

Importing Sales History

The `import_history` command imports sales history data into a demand model from a file. Full and incremental importing is supported.

For incremental importing, the following fields in the import file must be configured:

Field	Description
start-date	The start date of the incremental range, not the model start date. For example, if your model starts on 2006–02–01, and you want to update starting in June, then 2006–06–01 would be the correct start date.
count	The number of buckets between the incremental range start date and the end of the sales history data. For example, if monthly time buckets are being used, the start date is 2006–06–01, and the end of the sales history data is 2007–01–01, the count would be 7.

Usage

```
import_history model filename [-incremental]
```

Parameters

The `import_history` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import sales history.
`aggregation_hierarchy`	The aggregation hierarchy containing the sales history.
`filename`	The name of the file from which to import data. When specifying the filename, provide the path where the file resides.
`[-incremental]`	Specifies that the sales history data being imported is incremental.

Example

From the command prompt, type:

```
dm::transaction
 {import_history Model Default saleshistory_incremental.xml -incremental}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The import fails.

Copying Sales History to a Forecast Version

The `copy_sales_to_forecast_version` command copies sales history to a forecast version with the specified time lag.

This command is useful when rolling forward the model horizon and you want to populate a forecast with sales history. For example, when using monthly buckets you may want to use the `copy_sales_to_forecast_version` command to lag sales history by one or two years. It is recommended that the `balance` and/or `disaggregate` commands be used after the `copy_sales_to_forecast_version` command.

Usage

```
copy_sales_to_forecast_version model target_forecast_version
lag [aggregation_hierarchy] [demand_point_set] [number_of_buckets]
```

Parameters

The `copy_sales_to_forecast_version` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to copy sales history.
`target_forecast_version`	The demand model's target forecast version.
`lag`	The number of planning periods by which you want to lag the copied sales history.
`[aggregation_hierachy]`	The name of the aggregation hierarchy to copy.
`[demand_point_set]`	The name of the demand point set to copy. This parameter is option; if no demand point set is specified, all demand points are copied.
`[number_of_buckets]`	The number of time buckets that you want to copy to a forecast version. This parameter is optional; if no value is specified, the entire sales history is copied. For example, specifying three time buckets will copy the three most recent periods of sales history.

Example

From the command prompt, type:

```
dm::transaction
{copy_sales_to_forecast_version Bikes Reconciled 12 12}
```

Exceptions

This command throws an exception if:

- The model does not exist.
- The forecast version for the target demand model does not exist.

Range Profile Management

This section provides an overview of range profile management commands and discusses how to:

- List range profiles.
- Delete range profiles.
- Remove all range profile assignments.
- Remove a range profile assignments.
- Generate range profile alert reports.
- Delete range profile alert reports.

- Export range profile alert reports.
- Export range profiles.
- Import range profiles.
- Export specific range profile demand point assignments.
- Import specific range profile demand point assignments.
- Export all range profile demand point assignments.
- Import all range profile demand point assignments.

Understanding Range Profile Management Commands

Range forecasts enable you to compare multiple statements of demand across a forecasting horizon. You can generate optimistic, pessimistic, and most-likely forecasts, and you can compare and contrast them.

You can use these commands to manage range forecasts:

- `list_range_profiles`
- `delete_range_profile`
- `unassign_all_range_profiles`
- `unassign_range_profile`
- `generate_range_profile_alert_reports`
- `delete_range_profile_alert_reports`
- `export_range_profile_alert_reports`
- `import_range_profiles`
- `export_range_profiles`
- `import_demand_point_range_profiles_assignment`
- `export_demand_point_range_profiles_assignment`
- `import_all_demand_point_range_profiles_assignment`
- `export_all_demand_point_range_profiles_assignment`

Listing Range Profiles

The `list_range_profiles` command returns a list of all of the range profiles in a demand model.

Usage

```
list_range_profiles model
```

Parameters

The `list_range_profiles` command takes this parameter:

Parameter	Description
`model`	The name of the demand model that contains the range profiles that you want to list.

Example

From the command prompt, type:

```
dm::transaction
{list_range_profiles Sales}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profiles in the model.

Deleting Range Profiles

The `delete_range_profile` command removes a specific range profile in a demand model.

Usage

```
delete_range_profile model rangeProfile
```

Parameters

The `delete_range_profile` command takes these parameters:

Parameter	Description
`model`	The name of the demand model that contains the range profile that you want to delete.
`rangeProfile`	The name of the range profile that you want to delete.

Example

From the command prompt, type:

```
dm::transaction
{delete_range_profile Marketing Optimistic}
```

Exceptions

This command throws an exception if the demand model or the range profile does not exist.

Removing All Range Profile Assignments

The `unassign_all_range_profiles` command removes the assignment information for all range profiles in a demand model.

Usage

```
unassign_all_range_profiles model
```

Parameters

The `unassign_all_range_profiles` command takes this parameter:

Parameter	Description
model	The name of the demand model that contains the range profile assignment information that you want to remove.

Example

From the command prompt, type:

```
dm::transaction
 {unassign_all_range_profiles Marketing}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profiles in the model.

Removing a Range Profile Assignment

The unassign_range_profile command removes a range profile from a specific aggregation hierarchy in a demand model.

Usage

```
unassign_range_profile model aggregation_hierarchy rangeProfile
```

Parameters

The unassign_range_profile command takes these parameters:

Parameter	Description
model	The name of the demand model.
aggregation_hierarchy	The name of the aggregation hierarchy that contains the range profile with the assignment information that you want to remove.
rangeProfile	The name of the range profile.

Example

From the command prompt, type:

```
dm::transaction
{unassign_range_profile Sales "Product by Region" Pessimistic}
```

Exceptions

This command throws an exception if the demand model or the range profile does not exist.

Generating Range Profile Alert Reports

When there are demand points outside of a range profile, the application generates alerts that identify the demand points outside of the profile. The generate_range_profile_alert_reports command generates range profile alerts and saves them to the database.

Usage

```
generate_range_profile_alert_reports model
```

Parameters

The `generate_range_profile_alert_reports` command takes this parameter:

Parameter	Description
`model`	The name of the demand model for which you want to generate an alert report.

Example

From the command prompt, type:

```
dm::transaction
{generate_range_profile_alert_reports DurableGoods}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profiles in the model.

Deleting Range Profile Alert Reports

Range profile alert reports exist in the database. If you find that you need to delete all of the existing alert reports, use the`delete_range_profile_alert_reports` command.

Usage

```
delete_range_profile_alert_reports model
```

Parameters

The `delete_range_profile_alert_reports` command takes this parameter:

Parameter	Description
`model`	The name of the demand model containing the alert reports that you want to delete.

Example

From the command prompt, type:

```
dm::transaction {delete_range_profile_alert_reports SoftDrinks}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profile alert reports in the model.

Exporting Range Profile Alert Reports

You can export range profile alert reports from the database to an XML file using the `export_range_profile_alert_reports` command.

Usage

```
export_range_profile_alert_reports model filename
```

Parameters

The `export_range_profile_alert_reports` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the alert reports that you want to export.
`filename`	The name of the XML file into which the alert reports will be exported.

Example

From the command prompt, type:

```
dm::transaction
{export_range_profile_alert_reports BikeParts bp_alerts.xml}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profile alert reports in the model.

Exporting Range Profiles

You can export range profile definitions from the database to an XML file using the `export_range_profiles` command. The resulting XML file contains the following information:

- Then name of the range profiles.
- The units of measure used by the range profiles.
- The upper and lower limits of the flex fences in the range profiles.

Usage

```
export_range_profiles model filename
```

Parameters

The `export_range_profiles` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the range profiles that you want to export.
`filename`	The name of the XML file into which the range profile information will be exported.

Example

From the command prompt, type:

```
dm::transaction
 {delete_range_profile_alert_reports BikeParts bp_alerts.xml}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profile alert reports in the model.

Importing Range Profiles

You can import range profile definitions from an XML file into the database using the`import_range_profiles` command.

Usage

```
import_range_profiles model filename
```

Parameters

The `import_range_profiles` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the range profile information.
`filename`	The name of the XML that contains the range profile information.

Example

From the command prompt, type:

```
dm::transaction
 {import_range_profiles BikeParts bp_alerts.xml}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profile alert reports in the model.

Exporting Specific Range Profile Demand Point Assignments

Range profiles have demand points assigned to them. You can export range profile demand point assignments for a specific aggregation hierarchy from the database to an XML file using the `export_demand_point_range_profiles_assignment` command. The resulting XML file contains the following information:

- The names of the range profiles.
- The product, location, and channel paths of the assigned demand points.

Usage

```
export_demand_point_range_profiles_assignment model aggregation_hierarchy
filename
```

Parameters

The `import_range_profiles` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the demand point assignment information.
`aggregation_hierarchy`	The aggregation hierarchy containing the demand points that were assigned to the range profiles.
`filename`	The name of the XML that into which you will export the demand point assignment information.

Example

From the command prompt, type:

```
dm::transaction
 {export_demand_point_range_profiles_assignment CanadaEast Default dp_⇒
assignment.xml}
```

Exceptions

This command throws an exception if:

- The demand model does not exist.
- There are no range profiles in the model.
- You do not specify an aggregation hierarchy.

Importing Specific Range Profile Demand Point Assignments

You can import range profile demand point assignment information for a specific aggregation hierarchy from an XML file into the database using the `import_demand_point_range_profiles_assignment` command.

Usage

```
import_demand_point_range_profiles_assignment model aggregation_hierarchy
filename
```

Parameters

The `import_range_profiles` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the demand point assignment information.

Parameter	Description
aggregation_hierarchy	The name of the aggregation hierarchy that will contain the demand points that were assigned to the range profiles.
filename	The name of the XML that contains the demand point assignment information that you will import into the demand model.

Example

From the command prompt, type:

```
dm::transaction
 {import_demand_point_range_profiles_assignment CanadaEast Default dp_⇒
assignment.xml}
```

Exceptions

This command throws an exception if:

- The demand model does not exist.
- There are no range profiles in the model.
- You do not specify an aggregation hierarchy.

Exporting All Range Profile Demand Point Assignments

Range profiles have demand points assigned to them. You can export all of the range profile demand point assignments from the database to an XML file using the `export_all_demand_point_range_profiles_assignment` command. The resulting XML file contains the following information:

- The names of the range profiles.
- The product, location, and channel paths of the assigned demand points.

Usage

```
export_all_demand_point_range_profiles_assignment model filename
```

Parameters

The `import_range_profiles` command takes these parameters:

Parameter	Description
model	The name of the demand model from which you want to export the demand point assignment information.
filename	The name of the XML that into which you will export the demand point assignment information.

Example

From the command prompt, type:

```
dm::transaction
 {export_all_demand_point_range_profiles_assignment CanadaEast dp_assignment.xml}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profiles in the model.

Importing All Range Profile Demand Point Assignments

You can import range profile demand point assignment information from an XML file into the database using the `import_all_demand_point_range_profiles_assignment` command.

Usage

```
import_all_demand_point_range_profiles_assignment model filename
```

Parameters

The `import_range_profiles` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the demand point assignment information.
`filename`	The name of the XML that contains the demand point assignment information that you will import into the demand model.

Example

From the command prompt, type:

```
dm::transaction
{import_all_demand_point_range_profiles_assignment CanadaEast dp_assignment.xml}
```

Exceptions

This command throws an exception if the demand model does not exist or if there are no range profiles in the model.

Forecast History Management

This section provides an overview of forecast history management commands and discusses how to:

- Clear forecast history.
- Export forecast history data.
- Import forecast history data.

Understanding Forecast History Management Commands

You can import and export forecast history for existing forecast versions. Importing and exporting forecast history data enables you to view historical data and compare it to sales history for the same demand points. You can also clear forecast history data from a forecast version in a demand model.

You can use these commands to manage forecast history data:

- `clear_forecast_history`
- `export_forecast_history`
- `import_forecast_history`

Clearing Forecast History

The `clear_forecast_history` command clears the historical forecast data in the demand model.

Usage

```
clear_forecast_history model forecast_history
```

Parameters

The `clear_forecast_history` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to clear the forecast history.
`forecast_history`	The forecast version of the demand model from which you want to clear historical data.

Example

From the command prompt, type:

```
dm::transaction
 {clear_forecast_history Model Sales}
```

Exceptions

This command throws an exception if the demand model or the forecast version do not exist.

Exporting Forecast History Data

The `export_forecast_history` command exports the forecast history data from a demand model into an XML file.

Usage

```
export_forecast_version model forecast_history
unit_of_measure filename
```

Parameters

The `export_forecast_history` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export forecast history.

Parameter	Description
unit_of_measure	The unit of measure in which the forecast data will be exported.
filename	The name of the file into which you want to export data. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
 {export_forecast_history Sales dollars forecasts.xm}
```

Exceptions

This command throws an exception if the unit of measure does not exist in the model or if these items do not exist:

- Demand model.
- Historical forecast.
- Forecast version.

Importing Forecast History Data

After you have created a model, it is an empty demand model. The `import_forecast_history` command imports forecast history into a forecast version of the demand model. At any particular point, existing forecasts are either overridden by the new data (if the import contains a forecast for that point) or left unaltered (if the imported data does not include data for that point). When you import forecast history data that has missing values, the fields for the missing values are blank when you view the imported data.

Usage

```
import_forecast_history model forecast_history filename
```

Parameters

The `import_forecast_history` command takes these parameters:

Parameter	Description
model	The name of the demand model into which forecasts are imported.
forecast_history	The name of the forecast version that you want to import.
filename	The name of the file from which data is imported. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_forecast_history model Sales1 forecasts.xml}
```

Exceptions

This command throws an exception if the unit of measure does not exist in the model or if these items do not exist:

- Demand model.
- Historical forecast.
- Forecast version.

User Management

This section provides an overview of user management commands and discusses how to:

- Set administrator.
- Add a new user.
- Delete a user.
- Set a password.
- Manage user information.
- List users.

Understanding User Management Commands

DASH user management functions enable you to add, delete, and list users. They also enable you to set user passwords and permissions for reading, writing, and reconciling forecast data.

You can use these commands to manage users:

- `set_admin`
- `add_user`
- `delete_user`
- `password`
- `user_info`
- `list_users`
- `permissions`

Setting Administrator Privileges

The `set_admin` command uses a user name and an optional Boolean value. It is used to determine whether a user is an administrator, to change a user to an administrator, or to change an administrator to a user. This is useful because some commands are restricted to administrators.

This command is for existing users only. The `set_admin` command returns an answer of 1 (true) or 0 (false) depending on whether the user is the administrator.

Usage

```
set_admin user [true|false]
```

Parameters

The `set_admin` command takes these parameters:

Parameter	Description
`user`	The name of the user to which you want to grant administrator privileges, or whose status that you want to determine.

Example

When this command is typed at the command prompt, the `set_admin` command returns a value of 1 (true), meaning the specified user has administrator privileges:

```
% dm::transaction
{ set_admin admin} 1
```

When this command is typed at the command prompt, the `set_admin` command returns a value of 0 (false), meaning the specified user has administrator privileges:

```
% dm::transaction
{set_admin Bradley} 0
```

When this command is typed at the command prompt, the `set_admin` command assigns the specified user administrator privileges, and returns a value of 1 (true) confirming that administrator privileges have been granted:

```
% dm::set_admin Bradley true 1
```

Adding New Users

The `add_user` command creates a new user account, with no permissions for any forecasts.

Usage

```
add_user user password
```

Parameters

The `add_user` command takes these parameters:

Parameter	Description
`user`	The name for the new user.
`password`	The initial password for the user.

Exceptions

This command throws an exception if the user already exists or if you are not an administrator.

Deleting Users

The `delete_user` command deletes a user.

Usage

```
delete_user user
```

Parameters

The `delete_user` command takes this parameter:

Parameter	Description
`user`	The name of the existing user to delete.

Exceptions

This command throws an exception if the user does not exist.

Setting Passwords

The `password` command assigns a password to a user, or changes a user's password.

The manner in which this command is carried out depends on the number of arguments that are supplied. If the username is the only argument that is supplied for this command, the user's encrypted password is returned. Otherwise, the user's password is replaced with the new password. Unless you are an administrator, you can change only your own password. If you are an administrator, you can change anybody's password.

If you specify the `-crypted` parameter, then the password string that is passed to DASH is interpreted as having already been encrypted, and it is stored directly in the database.

Note. Administrators cannot read the passwords of the users since the passwords are stored only in encrypted form. However, administrators can read and set encrypted passwords for backup and data migration purposes.

Usage

```
password user [password [-crypted]]
```

Parameters

The `password` command takes these parameters:

Parameter	Description
`user`	The name of an existing user.
`password`	The initial password for the user.
`[-crypted]`	Indicates that the provided password is encrypted.

Exceptions

This command throws an exception for these conditions:

- The user does not exist.
- You do not have permissions to change passwords.

Managing User Information

The `user_info` command retrieves the information for a user from the database, or sets the specified information for a user. The manner in which this command is carried out depends on the number of arguments supplied. If you supply only a user's name, this command returns that user's information in this format:

```
full_name description email_address phone_number mobile_number fax_number
```

Note. If some of the parameters are not set, the `user_info` command returns "{}". If you provide all of the arguments, the command overrides the user's information. For example, when you specify the user's full name, description, email, telephone number, mobile telephone number, and fax number, the system overrides that information in the database.

Unless you are an administer, you can change only your own user information. If you are an administrator, you can change the information of any user.

Usage

```
user_info user [full_name description email phone mobile fax]
```

Parameters

The `user_info` command takes these parameters:

Parameter	Description
`user`	The name of an existing user.
`full_name`	The actual name of the user (for example Jason Carver).
`description`	The description of the user's job (for example, senior management).
`email`	The email address of the user.
`phone`	The telephone number of the user.
`mobile`	The mobile telephone number of the user.
`fax`	The fax number of the user.

Exceptions

This command throws an exception if the user does not exist.

Listing Users

The `list_users` command returns a list with the names of all users. This command is useful when you want to verify who has access to the application, and which users you need to remove.

Usage

```
list_users model
```

Parameters

The `list_users` command takes this parameter:

Parameter	Description
model	The name of the demand model containing the names of the users that you want to list.

Managing Permissions for Existing Users

The permissions command sets and displays the permissions that a user has for accessing a demand model. The manner in which this command is carried out depends on the number of arguments that are supplied. If you supply only the model, forecast version, and user name, the permissions command returns a list of permissions for the user for the forecast version in the model. Otherwise, this command sets the user's permissions for the forecast version in the model.

Usage

```
permissions model forecast_version user [{[read] [write] [reconcile]}]
```

Parameters

The permissions command takes these parameters:

Parameter	Description
model	The name of the demand model in which the user is going to participate.
forecast_version	The name of the instance to which the user is being given permissions to work.
user	The name of the user who will be participating.
read, write, reconcile	The specific permissions that the user is being given. Read permission makes the instance visible to the user. Write permission enables the user to edit forecast data or upload data into the forecast version. Reconcile permission enables the user to reconcile other forecasts into the forecast version.

Access Rights Management

This section provides an overview of access rights management commands and discusses how to:

- Import access rights.
- Export access rights.

Understanding Access Rights Management Commands

The access rights management commands enable you to grant and revoke user access rights to demand models.

You can use these commands to manage access user rights:

- import_access_rights
- export_access_rights

Importing Access Rights

The `import_access_rights` command imports access rights information from an XML file into the demand model. If an entry with the same user name exists in both the XML file and the demand model, this command overwrites the access rights information in the demand model with the information from the XML file.

The import process fails if any of this information that is in the XML file is not found in the demand model:

- User.
- Demand point set.
- Forecast version.

Note. You must have administrator privileges to run this command.

Usage

```
import_access_rights model filename
```

Parameters

The `import_access_rights` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the access rights information
`filename`	The name of the XML file containing the access rights information. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {import_access_rights ConferenceRoom BikesModel.xml}
```

Exceptions

This command throws an exception if the demand model does not exist.

Exporting Access Rights

The `export_access_rights` command exports access rights information from a demand model into an XML file.

Note. You must have administrator privileges to run this command.

Usage

```
export_access_rights model filename
```

Parameters

The `export_access_rights` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the access rights information.
`filename`	The name of the XML file into which you want to export the access rights information. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_access_rights TotalModel BikesModel.xml}
```

Exceptions

This command throws an exception if the model does not exist.

Accuracy Report Management

This section provides an overview of accuracy report management commands and discusses how to:

- Generate an accuracy report.
- Generate accuracy reports for weighting categories.
- Import an accuracy report.
- Export an accuracy report.

Understanding Accuracy Report Management Commands

The accuracy report commands enable you to create, import, and export reports that contain accuracy measurements for individual forecasts. You can use these commands to manage accuracy reports:

- `report_accuracy`
- `generate_category_accuracy_report`
- `import_accuracy_report`
- `export_accuracy_report`

Generating Accuracy Reports

The `report_accuracy` command to generate a report containing accuracy measures for individual forecasts.

Usage

```
report_accuracy model aggregation_hierarchy report_name filename
```

Parameters

The `report_accuracy` command takes these parameters:

Parameter	Description
`model`	The name of the demand model for which you want to generate an accuracy report.
`aggregation_hierarchy`	The name of the aggregation hierarchy.
`filename`	The name of the XML or CSV file containing the generated report. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
 {report_accuracy ConferenceRoom ProductGroupA accuracyreport.xml}
```

Exceptions

The command throws an exception for these conditions:

- The demand model does not exist.
- The import fails for another reason.

Generating Accuracy Reports for Weighting Categories

The `generate_category_accuracy_report` command generates a report containing all accuracy measurements and tracking signals for weighting categories.

Usage

```
generate_category_accuracy_report model start_period end_period filename
```

Parameters

The `generate_category_accuracy_report` command takes these parameters:

Parameter	Description
`model`	The name of the demand model
`start_period`	The index of the first time bucket to use in the calculation, starting at 0.
`end_period`	The index of the last time bucket to use in the calculation, starting at 0.
`filename`	The name of the XML or CSV file that will contain the generated report. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
```

```
{generate_category_accuracy_report ConferenceRoom 0 4 accuracyreport.csv}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The demand model has an invalid starting period, ending period, or both.
- The filename has the wrong extension.

Importing Accuracy Reports

The `import_accuracy_report` command imports an accuracy report into a demand model. When specifying the file name, give the path where the file resides.

Usage

```
import_accuracy_report model filename
```

Parameters

The `import_accuracy_report` command takes these parameters:

Parameters	Description
`model`	The name of the demand model into which you want to import the accuracy report.
`aggregation_hierarchy`	The name of the aggregation hierarchy.
`filename`	The name of the file from which to import the accuracy report data. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_accurancy_report ConferenceRoom ProductGroupA accuracyreport.xml}
```

Exceptions

The command throws an exception for these conditions:

- The demand model does not exist.
- The hierarchy does not exist.
- The import fails for another reason.

Exporting Accuracy Reports

The `export_accuracy_report` command exports accuracy report settings to an XML file.

Usage

```
export_accuracy_report model hierarchy filename
```

Parameters

The `export_accuracy_report` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export the accuracy report.
`aggregation_hierarchy`	The name of the aggregation hierarchy.
`filename`	The name of the file into which data is exported. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction {
export_accuracy_report ConferenceRoom ProductGroupA accuracyreport.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The hierarchy does not exist.
- An invalid unit of measure is specified.
- The system fails to export the settings into the file.

Demand Point Sets Management

This section provides an overview of demand point sets management commands and discusses how to:

- Import demand point sets.
- Export demand point sets.
- List demand point sets.
- Refresh multiple demand point sets.
- Refresh a single demand point sets.

Understanding Demand Point Sets Management Commands

Demand point sets are groups of demand points that are tailored to the specific needs of a group of users. Demand point sets are both a filtering and a security mechanism. Users associated with a particular demand point set can view and edit only the demand points for which they are responsible, saving them the trouble of having to search through every demand point in a model. In addition, users can only view the demand points to which they have access.

You can use these commands to manage demand point sets:

- import_demand_point_sets
- export_demand_point_sets
- list_demand_point_sets
- refresh_demand_point_sets
- refresh_demand_point_set

See Also

Chapter 17, "Using the Demand Automation Shell," Access Rights Management, page 212

Importing Demand Point Sets

The import_demand_point_sets command imports one or more demand point sets from an XML file into an existing demand model. This command does not delete the demand points sets that exist in the demand model. However, the command overwrites any matching demand points sets in the demand model.

Note. You must have administrator privileges to run this command.

Usage

```
import_demand_point_sets model aggregation_hierarchy filename
```

Parameters

The import_demand_point_sets command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import the demand point sets.
aggregation_hierarchy	The name of the aggregation hierarchy for the demand point set.
filename	The name of the XML file containing the demand point sets. When specifying the filename, provide the path where the file resides.

Example

From the command prompt, type:

```
dm::transaction
{import_demand_point_sets TotalModel TotalSales BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist in the model.

Exporting Demand Point Sets

The `export_demand_point_sets` command exports all of the demand point sets in a demand model into an XML file.

Note. You must have administrator privileges to run this command.

Usage

```
export_demand_point_sets model aggregation_hierarchy filename
```

Parameters

The `export_demand_point_sets` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the demand point sets that you want to export.
`aggregation_hierarchy`	The name of the aggregation hierarchy for the demand point set.
`filename`	The name of the XML file into which you want to export the demand point sets. When specifying the filename, provide the path where the file should reside.

Example

From the command prompt, type:

```
dm::transaction
{export_demand_point_sets TotalModel TotalSales BikesModel.xml}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist in the model.

Listing Demand Point Sets

The `list_demand_point_sets` command returns a list that contains the names of all demand point sets to which the system administrator has permissions in the current model and aggregation hierarchy.

Usage

```
list_demand_point_sets model aggregation_hierarchy
```

Parameters

The `list_demand_point_sets` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the demand point sets that you want to list.
aggregation_hierarchy	The name of the aggregation hierarchy for the demand points listed.

Example

From the command prompt, type:

```
dm::transaction {
list_demand_point_sets ConferenceRoom StateSales}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist in the model.

Refreshing Multiple Demand Point Sets

The `refresh_demand_point_sets` command refreshes the list of demand point sets to which the system administrator has permissions. Refreshing demand point sets ensures that rules are applied against any newly created demand points.

Usage

```
refresh_demand_point_sets model aggregation_hierarchy
```

Parameters

The `refresh_demand_point_sets` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the demand point sets that you want to refresh.
aggregation_hierarchy	The name of the aggregation hierarchy for the demand points listed.

Example

From the command prompt, type:

```
dm::transaction
 {refresh_demand_point_sets Bikes StateSales}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist in the model.

The system will also display a warning message if any demand point set contains rules that are invalid.

Refreshing Single Demand Point Sets

The `refresh_demand_point_set` command refreshes a specific demand point set to which the system administrator has permission. Refreshing a demand point set ensures that rules are applied against a specific demand point set.

Usage

```
refresh_demand_point_set model aggregation_hierarchy demand_point_set
```

Parameters

The `refresh_demand_point_set` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the demand point set that you want to refresh.
`aggregation_hierarchy`	The name of the aggregation hierarchy for the demand point set listed.
`demand_point_set`	The name of the demand point set that is being refreshed.

Example

From the command prompt, type:

```
dm::transaction {refresh_demand_point_set Bikes StateSales Marketing}
```

Exceptions

This command throws an exception for these conditions:

- The demand model does not exist.
- The aggregation hierarchy does not exist in the model.

Fixed Points Management

This section provides an overview of fixed point management commands and discusses how to:

- Set the fixable aggregation hierarchy.
- Import fixed points.
- Export fixed points.
- Clear fixed points.

Understanding Fixed Point Management Commands

Sometimes demand is fixed for certain demand points in some periods within forecast versions in a model. Using DASH commands, it is possible to set the fixable hierarchy, import, export, and clear fixed points based on the contents of an XML file. In addition, after the demand has been fixed, the model can be disaggregated and/or balanced automatically. Notes can be appended.

You can use these commands to manage fixed points in your demand model:

- `fixing_hierarchy`
- `import_fixed_points`
- `export_fixed_points`
- `clear_fixed_points`

Setting Fixable Aggregation Hierarchy

The `fixing_hierarchy` command sets the fixable aggregation hierarchy in the model. Only one aggregation hierarchy in a demand model can be fixable at a time.

Usage

```
fixing_hierarchy model aggregation_hierarchy
```

Parameters

The `fixing_hierarchy` command takes these parameters:

Parameter	Description
`model`	The name of the demand model in which you want to set the fixable aggregation hierarchy.
`aggregation_hierarchy`	The name of the fixable aggregation hierarchy.

Example

From the command prompt, type:

```
dm::transaction
{fixing_hierarchy Widgets RegionalSales}
```

Exceptions

If the demand model is the only argument supplied, this command queries the model and returns the name of the aggregation hierarchy that is the fixing hierarchy in the following format: `fixing_aggregation_hierarchy`. If you provide the second argument, `aggregation_hierarchy`, it sets that aggregation hierarchy as the fixing hierarchy.

Importing Fixed Points

The `import_fixed_points` command imports fixed point information from an XML file into a demand model.

Usage

```
import_fixed_points model aggregation_hierarchy forecast_version filename
[-disaggregation_method] [-append_notes] [-balance_above]
```

Parameters

The `import_fixed_points` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import fixed point information.
`aggregation_hierarchy`	The name of the aggregation hierarchy into which you want to import fixed point information. If the hierarchy specified is not the "fixing" hierarchy, an error is displayed.
`forecast_version`	The name of the forecast version into which you want to import fixed point information. You can import fixed point information to a different forecast version than that previously exported.
`filename`	The name of the XML file containing fixed point information.
`-disaggregation_method`	The type of disaggregation to perform before fixing the demand points. Options are: • `-system_profile` Disaggregates the forecast based on a system-defined pattern. Leaf-level demand points are copied from the system profile into the current profile and the existing demand points are ignored. You can use this information if little or no data is at the lowest level of the current forecast version. Another reason to use the system profile is for low-level data that cannot be forecasted well. For example, the usage of a distribution pattern based upon average sales history rather than statistical forecasts of size and color combinations in the apparel sector. • `-top_down` Distributes the forecast data downwards from the highest aggregation level according to the underlying pattern in the current forecast version. The value in each time bucket adds up to the value of the parent forecast. • `-bottom_up` Distributes forecast data upwards from the lowest aggregation level according to the underlying pattern in the current forecast version. Use this method when low-level forecasts have proven to be reliable.
`-append_notes`	An optional parameter that automatically appends notes to forecasts during the disaggregation process. **Note.** This parameter is only available when using the -top_down disaggregation method.
`-balance_above`	Specifies whether to balance demand points above the fixed demand point.

Example

```
dm::transaction
{import_fixed_points Widgets RegionalSales Sales widgets_sales.xml -system_profile
 -balance_above}
```

Exceptions

This command throws an exception if the aggregation hierarchy is not the “fixing” hierarchy.

Exporting Fixed Points

The `export_fixed_points` command saves fixed point information to an XML file

Usage

```
export_fixed_points model
aggregation_hierarchy forecast_version
unit_of_measure filename
```

Parameters

The `export_fixed_points` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export fixed demand point information.
`aggregation_hierarchy`	The name of the aggregation hierarchy containing the fixed points you want to export.
`forecast_version`	The name of the forecast version whose fixed points you want to export.
`unit_of_measure`	The unit of measure used by the forecast version whose fixed points you want to export.
`filename`	The name of the XML file to which the fixed point information is exported.

Example

From the command prompt, type:

```
dm::transaction
{export_fixed_points Widgets RegionalSales Sales Each widgets_⇒
sales.xml}
```

Example

This command throws an exception if the aggregation hierarchy is not the “fixing” hierarchy.

Clearing Fixed Points

The `clear_fixed_points` command clears specific fixed points in the demand model based on the contents of an XML file.

Usage

```
clear_fixed_points model aggregation_hierarchy forecast_version filename
```

Parameters

The `clear_fixed_points` command takes these parameters:

Parameter	Description
`model`	The name of the demand model whose fixed points you want to clear.
`aggregation_hierarchy`	The name of the aggregation hierarchy containing the fixed points you want to clear. If this hierarchy is not the "fixing" hierarchy, an error is displayed.
`forecast_version`	The name of the forecast version whose fixed points you want to clear. The forecast version does not need to be the same as that originally exported.
`filename`	The name of the XML file that specifies the demand points to clear. If no filename is specified, all fixed points are cleared.

Example

From the command prompt, type:

```
dm::transaction
{clear_fixed_points Widgets RegionalSales Sales widgets_sales.xml}
```

Exceptions

This command throws an exception if the aggregation hierarchy is not the "fixing" hierarchy.

Managing Units of Measure Assignment

This section provides an overview of units of measure assignment management commands and discusses how to:

- Import units of measure assignments.
- Export units of measure assignments.

Understanding Managing Units of Measure Assignment Commands

Because a single unit of measure may not be applicable to all of the items across a product line, you can assign units of measure to individual products and product groups at the leaf demand point level.

You can use these commands to manage units of measure assignment:

- `import_units_of_measure_assignment`
- `export_units_of_measure_assignment`

See Also

Chapter 11, "Working with Units of Measure," Assigning Units of Measure to Demand Points, page 70

Importing Units of Measure Assignments

The `import_units_of_measure_assignment` command imports unit of measure assignment information from an XML file into a demand model.

Usage

```
import_units_of_measure_assignment model aggregation_hierarchy filename
```

Parameters

The `import_units_of_measure_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model into which you want to import the units of measure assignment information.
`aggregation_hierarchy`	The name of the aggregation hierarchy into which the units of measure assignment information that you want to import.
`filename`	The name of the XML file containing the units of measure assignment information.

Example

From the command prompt, type:

```
dm::transaction
{import_units_of_measure_assignment Widgets RegionalSales widgets_sales.xml}
```

Exceptions

This command throws an exception if the demand model or aggregation hierarchy do not exist.

Exporting Units of Measure Assignments

The `export_units_of_measure_assignment` command saves unit of measure assignment information to an XML file.

Usage

```
export_units_of_measure_assignment model aggregation hierarchy filename
```

Parameters

The `export_units_of_measure_assignment` command takes these parameters:

Parameter	Description
`model`	The name of the demand model from which you want to export unit of measure assignment information.

Parameter	Description
aggregation_hierarchy	The name of the aggregation hierarchy containing the units of measure assignment information that you want to export.
filename	The name of the XML file into which the units of measure assignment information will be exported.

Example

From the command prompt, type:

```
dm::transaction
{export_units_of_measure_assignment Widgets RegionalSales
widgets_sales.xml}
```

Exceptions

This command throws an exception if the demand model or aggregation hierarchy do not exist.

Query Management

This section provides an overview of query management commands and discusses how to:

- Export queries.
- Import queries.
- List queries.
- Rename queries.
- Delete queries.
- Execute queries.
- Disaggregate queries.
- Export query forecasts.
- Export query sales histories.

Understanding Query Management Commands

Queries gives you the ability to search the database for items using any criteria. The queries perform a search is based on individual properties that, while they exist in the demand model, may not exist*together* in an aggregation hierarchy.

You can use these commands to manage queries:

- export_queries
- import_queries
- list_queries
- rename_queries
- delete_queries

- execute_queries
- disaggregate_query
- export_query_forecast
- export_query_results

See Also

Working with Ad Hoc Queries, *EnterpriseOne Supply Chain Planning Demand Management 9.0 Forecast Studio Implementation Guide*

Exporting Queries

The export_queries command exports all of the queries from a demand model into an XML file.

Usage

```
export_queries model filename
```

Parameters

The export_queries command takes these parameters:

Parameter	Description
model	The name of the demand model containing the queries that you want to export.
filename	The name of the XML file into which you want to export the queries.

Example

From the command prompt, type:

```
dm::transaction
{import_queries Widgets widgets_sales.xml}
```

Exceptions

This command throws an exception for the following conditions:

- The demand model does not exist.
- Any of the queries in the input file already exist in the demand model.

Importing Queries

The import_queries command imports query information that was exported into an XML file back into the database.

Usage

```
import_queries model filename
```

Parameters

The import_queries command takes these parameters:

Parameter	Description
model	The name of the demand model into which you want to import queries.
filename	The name of the XML file containing the query parameters.

Example

From the command prompt, type:

```
dm::transaction
{import_queries Widgets widgets_sales.xml}
```

Exception

This command throws an exception if the demand model does not exist.

Listing Queries

The `list_queries` command returns an alphabetically-sorted list of all the queries in the database. You can use this command to:

- Capture a list of the queries, which you can then use in a batch job to run multiple queries.
- Keep track of the queries in the database.

Usage

```
list_queries model
```

Parameters

The `list_queries` command takes this parameter:

Parameter	Description
model	The name of the demand model containing the queries that you want to list.

Example

From the command prompt, type:

```
dm::transaction
{list_queries Widgets}
```

Exceptions

This command throws an exception if the demand model does not exist.

Renaming Queries

The `rename_query` command enables you to change the names of queries to ones that are more descriptive.

Usage

```
rename_query model from_query to_query
```

Parameters

The `rename_query` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the query that you want to rename.
`from_query`	The current name of the query.
`to_query`	The new name for the query

Example

From the command prompt, type:

```
dm::transaction
 {rename_query TotSal04 TotalSalesApril}
```

Exceptions

This command throws an exception for the following conditions:

- The demand model does not exist.
- The query does not exist.
- An existing query has the name that you want to use to rename a query.

Deleting Queries

The `delete_query` command deletes queries that you no longer use, or which are invalid.

Usage

```
delete_query model query
```

Parameters

The `delete_query` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the query that you want to delete.
`query`	The name of the query that you want to delete.

Example

From the command prompt, type:

```
dm::transaction
{delete_query Widgets March_Sales}
```

Exceptions

This command throws an exception if the following do not exist:

- The demand model.

- The query.

Executing Queries

The `execute_query` command runs queries and writes the results to the database.

Usage

```
execute_query model query
```

Parameters

The `execute_query` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the query that you want to execute.
`query`	The name of the query to execute.

Example

From the command prompt, type:

```
dm::transaction
{execute_query Widgets TotalSales}
```

Exceptions

This command throws an exception for the following conditions:

- The demand model does not exist.
- The query does not exist.
- The query is invalid. A query can be invalid a unit of measure has been deleted, or if it contains an invalid property name.

Disaggregating Queries

The `disaggregate_query` command disaggregates the specified query forecasts.

Usage

```
disaggregate_query model query forecast_version_list [-system_profile]
[-only_modified_forecasts]
```

Parameters

The `disaggregate_query` command takes these parameters:

Parameter	Description
`model`	The name of the demand model containing the query forecasts that you want to disaggregate.
`query`	The name of the query that you want to disaggregate.
`forecast_version_list`	A list of forecast versions that you want to disaggregate.

Parameter	Description
[-system_profile]	An optional parameter specifying the profile you want to use. If this parameter is specified, the system profile is used as the disaggregation profile. If the parameter is not included, the forecast profile is used.
[-only_modified_forecasts]	An optional parameter specifying to only disaggregate forecasts that have been modified since the last query execution. If the parameter is not included, every forecast is disaggregated.

Example

From the command prompt, type:

```
dm::transaction
{disaggregate_query Widgets TotalSales {Sales Marketing}}
```

Exceptions

This command throws an exception if one of the following conditions is met:

- The model does not exist.
- The query does not exist.
- The query is not valid.
- The forecast version list is not valid for the query.

Exporting Query Forecasts

The `export_query_forecasts` command exports the forecasts generated after a query is run into either an XML or a CSV file. When you export query forecasts, you are writing both the query parameters and the latest set of query results to the file.

Note. You cannot export query forecasts if a query has not been run.

Usage

```
export_query_forecasts model query forcast_version_list filename -xml|-csv
```

Parameters

The `export_query_forecasts` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the query forecasts that you want to export.
query	The name of the query containing the information that you want to export.
forecast_version_list	A list of the forecast versions that will be exported with the query parameters. The list is enclosed in curled braces {} and each forecast version in the list is separated by a space.

Parameter	Description
filename	The name of the XML or CSV file into which the query results will be exported.
-xml|-csv	Specifies the format of the file into which the query results will be exported.

Example

From the command prompt, type:

```
dm::transaction
 {export_query_results Widgets {Sales Marketing Statistical} widgets_sales.xml -⇒
xml}
```

Exceptions

This command throws an exception if the following do not exist:

- The demand model.
- The query.
- One or more forecast versions.

Exporting Query Sales Histories

The `export_query_sales_history` command exports the sales histories in a query to either an XML or a CSV file. The file contains this information:

- The query properties by which the query results are aggregated.
- The date.
- The value of the forecast on a particular date.

Note. You cannot export query forecast sales histories if a query has not been run.

Usage

```
export_query_sales_history model query filename -xml|-csv
```

Parameters

The `export_query_sales_history` command takes these parameters:

Parameter	Description
model	The name of the demand model containing the query sales history information that you want to export.
query	The name of the query containing the sales history information that you want to export
filename	The name of the XML or CSV file into which the sales history information will be exported.
-xml|-csv	Specifies the format of the file into which the query sales history will be exported.

Example

From the command prompt, type:

```
dm::transaction {
export_query_sales_history Widgets -csv widgets_sales.csv}
```

Exceptions

This command produces an exception if the demand model does not exist.

Backing Up Application Data

This section provides prerequisites for backing up data and discusses:

- Backup demand point data.
- Backup commands for demand models.

Prerequisites

You can backup application data by using the `dm_backup` script. Verify that these actions and items are available before you begin a backup:

- The `DC_HOME` environment variable is set to:
- Windows: `DC_HOME\bin`
- UNIX: `DC_HOME/bin`
- The database has been created.
- A valid database is available to write to with enough space to write the backup to.
- A valid directory path name for output is available.
- Write permissions for the target are set up.

Backing Up Demand Point Data

You can back up all demand point data in the database using the `dm_backup` script on UNIX or the `dm_backup.bat` batch file in Windows. The `dm_backup` command saves a copy of the data to the directory that you specify on the command line.

Usage

```
dm_backup [output]
```

Parameters

The `dm_backup` command takes these parameters:

Parameter	Description
[output]	This parameter represents the target backup directory. The default value is the current directory. For example in Windows navigate to c:\SCP\9.0\bin\ and then run dm_backup.bat. When the backup command finishes running, the current date is automatically added to the output filename - for example, Backup.2005-05-28.

Note. The output directory can be any absolute or relative path, including shared directories on any machine on a local network that is accessible from the server.

Example

From the Windows command prompt, type:

```
dm_backup c:\data\100104
```

From the UNIX command prompt, type:

```
dm_backup /home/backup/100104
```

Backup Commands for Demand Models

After you run the backup command, the system creates these files:

File Name	Description
dm_restore.bat (for Windows) dm_restore (for Unix)	The user must run this file for the restoration process.
restore.tcl (for Windows and Unix)	Contains the main Demand Automation Shell Tcl scripts. Called by dm_restore.bat (or dm_restore for Unix environments).
create_users.tcl (for Windows and Unix)	DASH Tcl script that stores and restores user profile information including passwords. This script is invoked from the main script restore.tcl, and called by dm_restore.bat (or dm_restore, for Unix environments).

The system creates a directory for each demand model. The directory uses the same name as the demand model and is structured like this example:

File Name	Description
Bikes	This directory holds all of the files associated with this demand model.

Restoring Data

This section discusses how to restore data.

Restoring Data

You can use the `dm_restore` script to restore any data that you have backed up. You can run this command if the database becomes corrupted or if users have added too much superfluous data to the database.

To run the command in Windows or UNIX, navigate to the backup output directory that was created by the backup process. You must provide the full path where is installed.

Usage

```
dm_restore [output]
```

Parameters

The `dm_restore` command has this parameter:

Parameter	Description
`[output]`	The directory into which the information will be restored. You must specify the directory in which the application is installed.

Example

From the Windows command prompt, navigate to the backup directory and type:

```
dm_restore c:\scp\9.0\DM
```

From the UNIX command prompt, navigate to the backup directory and type:

```
dm_restore /opt/scp/9.0/DM
```

Note. The database is not emptied prior to restore process. Only the data stored during the backup process is added (overridden). The restore process is executed in one transaction. If the transaction fails for any reason, the changes are rolled back.

CHAPTER 18

Importing Enterprise Data From EnterpriseOne Supply Chain Business Modeler

This chapter provides an overview of importing data from EnterpriseOne Supply Chain Business Modeler, prerequisites, and discusses how to:

- Create and modify model generation scenarios using Supply Chain Business Modeler.
- Work with the connector.
- Create an empty demand model.
- Work with connector packages.
- Use the refresh command.
- Use the publish command.

Understanding Importing Data From EnterpriseOne Supply Chain Business Modeler

You can import data from EnterpriseOne Supply Chain Management and Supply Chain Planning applications into Demand Forecasting using EnterpriseOne Supply Chain Business Modeler.

Supply Chain Business Modeler is a central data warehouse that enables you to convert supply chain data to the representation required by EnterpriseOne Demand Forecasting and other supply chain systems. After importing enterprise data into Supply Chain Business Modeler and converting it to the required representation, you can export the data for use in Demand Forecasting.

To specify how Demand Forecasting uses data exported from Supply Chain Business Modeler, you can create and run Supply Chain Business Modeler model generation scenarios. When you create a model generation scenario, you can select options for creating the data model in Demand Forecasting. When you run a model generation scenario, Supply Chain Business Modeler creates a model generation rules (.mgr) file that the connector uses when creating a Demand Forecasting model with the data.

Prerequisites

Before you can create, modify, or run model generation scenarios in Supply Chain Business Modeler, you must:

- Ensure that an extract area and at least one data folder exists for saving data files that you export from Supply Chain Business Modeler.

See Creating Extract Areas and Data Folders, Setting Up Extract Areas and Data Folders, *EnterpriseOne Supply Chain Business Modeler 9.0 Implementation Guide.*

- Save any Tcl scripts that you want to run as pre-export or post-export scripts in the Scripts folder of the extract area.

  See Scripts Folders, Setting Up Extract Areas and Data Folders, *EnterpriseOne Supply Chain Business Modeler 9.0 Implementation Guide.*

Windows Used to Create, Modify, and Run Model Generation Scenarios in Supply Chain Business Modeler

Form Name	Navigation	Usage
Add Model Generation Scenario	In Supply Chain Business Modeler, click the Model Generation Scenarios button in the Shortcuts bar. Select File, Add Model Generation Scenario. Click Next.In Supply Chain Business Modeler, click the Model Generation Scenarios button in the Shortcuts bar. Right-click the Model Generation Scenario Navigator and select Add Model Generation Scenario. Click Next.	Create model generation scenarios.
Modify Scenario	Start Supply Chain Business Modeler. Click the Model Generation Scenarios button in the Shortcuts bar. In the Model Generation Scenario Navigator, right-click the model generation scenario and select Modify Scenario. Click Next.	Modify model generation scenarios.
Generate Scenario	• Click the Model Generation Scenarios button in the Shortcuts bar. Select the model generation scenario that you want to run. Select File, Run. • Click the Model Generation Scenarios button in the Shortcuts bar. Right-click the model generation scenario that you want to run, and select Run.	Run model generation scenarios.

Creating and Modifying Model Generation Scenarios Using Supply Chain Business Modeler

Access the Add Model Generation Scenario window or the Modify Model Generation Scenario window.

Scenario Definition

Name Enter a name for the model generation scenario. The resulting model generation rules file will have the same name as the model generation scenario, with the .mgr extension. For example, if the model generation scenario name is scenario1, the resulting rules file will be scenario1.mgr.

Model Enter the name of the model from which the data was exported.

Data Location

Extract Area Enter the name of the extract area where the exported data packages are saved. You cannot create a model generation scenario an extract area exists.

Data Folder Select the name of the data folder where the exported data for the model generation scenario is saved. The model generation rules (.mgr) file will be created in this data folder when you run the model generation scenario.

Pre and Post Scripts

Pre-script Enter the name of a Tcl script that you want to run after the model generation scenario. Pre-scripts enable you to call external processes that you want to link to the model generation scenario. The Tcl script must be saved with the .tcl extension in the Scripts directory of the extract area used in the scenario, and cannot include Business Modeler Shell commands.

Post-script Enter the name of a Tool Command Language (Tcl) script that you want to run after the model generation scenario. Post-scripts enable you to call external processes that you would like to link to the model generation scenario. The Tcl script must be saved with the .tcl extension in the Scripts directory of the extract area used in the scenario, and cannot include Business Modeler Shell commands.

Application Integration Profile

Application integration profile Select the name of the Supply Chain Planning application where a model will be generated using the exported data.

Model Generation Details

Reduce Master List Values are:

Y: reduces the master list of item, branch, and channels, based on the demand points.

*N:*imports all items, branches, and channels.

Leaf Demand Point Location Selects where demand points are sourced from and can be sourced from the base or demand point history package. Values are:

Base

DemandPointHistory

Units of Measure Details

Currency The value for base currency in the demand model. There can only be one base currency.

Units of Measure List of all units of measure to be included in the demand model. If there is more than one unit of measure, use a space delimited list.

Item Details

Item Name Mapping Field selected to map to Name in Demand Management.

Values are:

Item Code

Item Name

Description

Alternate Item ID

Item Name *N:* do not include this field in Demand Management.

Y: include this field in Demand Management.

Alternate Item ID Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Description Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Planning UoM Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Shipping UoM Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Item Weight Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Weight UoM Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Volume Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Volume UoM Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Storage Requirements Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Location Details

Location Name Mapping Field selected to map to Name in Demand Management.

Values are:

Location Code

Location Name

Description

Alternate Branch ID

Branch Name Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Alternate Branch ID Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Branch Type Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Description Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

City Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

State/Province Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Country Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Postal Code Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Latitude Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Longitude Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Shipping Calendar Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Receiving Calendar Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Production Calendar Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Channel Details

Channel Name Mapping Field selected to map to Name in Demand Management.

Values are:

Channel Code

Channel Name

Description

Alternate Branch ID

Channel Name Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Channel Type Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Description Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

City Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

State/Province Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Country Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Postal Code Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Latitude Values are:

*N:*do not include this field in Demand Management.

Y: include this field in Demand Management.

Longitude Values are:

N: do not include this field in Demand Management.

Y: include this field in Demand Management.

Forecast Version Details

Forecast Versions List of forecast versions included in the demand model. If there is more than one forecast version, use a space delimited list.

Working with the Connector

Because different organizations often model their enterprise information in different ways, these variations in structure make data integration difficult. The application uses specific attributes in the demand model structure to store forecast data. Data from enterprise resource planning (ERP) applications might not fit into the structure of the demand model. The Connector helps to coordinate data that is created from different source applications and demand models.

EnterpriseOne Supply Chain Business Modeler is a centralized database of business data that is mapped to different business processes. The Connector is a EnterpriseOne Supply Chain Business Modeler driver that directly reads and writes EnterpriseOne Supply Chain Business Modeler XML documents to and from the database. Using only two commands, you can transfer data between EnterpriseOne Supply Chain Business Modeler and demand models.

Data transfer is versatile, because you can choose the EnterpriseOne Supply Chain Business Modeler data packages that you want to transfer to the demand model. You can also rename data attributes so that they conform to the demand model standards. This flexibility is helpful when you view and configure details of the transferred data in the demand model.

Creating an Empty Demand Model

This section discusses how to create an empty demand model.

After you run the model generation in EnterpriseOne Supply Chain Business Modeler with the demand model extract, you run the createModel command to generate an empty demand model. The empty demand model contains the following information:

- The sales history horizon.
- The appropriate units of measure.
- Product, location, and channel properties.
- One or more forecast versions.

To create an empty demand model:

1. At the command prompt, navigate to the start directory (for example, *c:/scp/9.0/common/start)* where the application is installed.
2. Type `run_dm_dm_connector` and then press Enter.
3. Enter the following command:

   `dm::model createModel` *<data folder> <model name> <model generation scenario file>*

   Where:

   - *<data folder>* is the path to the folder containing the EnterpriseOne Supply Chain Business Modeler data that will be the basis for the demand model.
   - *<model name>* is the name for the demand model.
   - *<model generation scenario file>* is the model generation file created by the EnterpriseOne Supply Chain Business Modeler.

Working with Connector Packages

This section provides an overview of the connector packages, and discusses how to refresh the following packages:

- Base.

- TimeSeries.
- DemandPointHistory.

Understanding Connector Packages

The Connector packages that are transferred from EnterpriseOne Supply Chain Business Modeler are used by the Connector to populate a working demand model with data. The data that is contained in EnterpriseOne Supply Chain Business Modeler data packages are mapped to coordinate attributes in the demand model through the use of the model generation rules file.

You can specify which data is transferred to the demand model by selecting the data packages to be transferred. Refreshing the entire demand model by using all of the data packages (except for the Enterprise Forecast package) transfers all of the data to the demand model. Refreshing specific demand model data, using any of the data packages, transfers specific data to the demand model.

The Enterprise Forecast package is the only package that transfers enterprise data from the demand model to EnterpriseOne Supply Chain Business Modeler.

Note. Packages can be in XML or gzip format.

Refreshing the Base Package

These attributes are updated from the base package when the Refresh command is executed:

Attribute	Description
Coordinates (product, location)	• A location or product property is updated with content from the base.xml file if it already exists. • The system identifies new locations and product properties which are created using data in the base.xml file. • A property that has a value of missing is given to an empty string or a missing property value. • An exception is thrown if the demand model does not contain the properties specified in the config.xml file.
Demand Points	If the demand model does not contain the demand point specified in the base.xml file, the demand point is added to the demand model with a description indicating that it is new.
Unit of Measure Conversion	The base package contains unit of measures conversions that do not contain effective dating. When refreshing data, the Connector applies unit of measure conversion rates for the first bucket of the sales history horizon, for all demand points in the demand model, for a specified product. The system ignores conversions to units of measure that are not used in the demand model.

Refreshing the TimeSeries Package

When you update the TimeSeries package, the system transfers multiple forecasts for valid product, location, and channel combinations for the specified forecast horizon. This information is used to generate a forecast version time series for a demand model.

Refreshing the DemandPointHistory Package

When you refresh the DemandPointHistory package, the system updates the sales history data for valid product, location, and channel combinations for the specified historical horizon. Sales history is used to determine forecast accuracy by comparing historical data with forecast data that is projected for the future. You might want to update this data frequently to ensure that forecast data is accurate and current.

Using the Refresh Command

This section provides an overview of the refresh command and discusses how to use the refresh command.

Overview of the Refresh Command

When you run the refresh command, the system transfers data in XML format from EnterpriseOne Supply Chain Business Modeler into the database. There are four commands which transfers all of the connector packages to the database and updates the entire demand model. You can also run one of the commands to update specific coordinate attributes and data into the database.

To use the refresh command:

1. At the command prompt, navigate to the start directory (for example, *c:/scp/9.0/common/start)* where the application is installed.
2. Type `run_dm_dm_connector` and then press Enter.
3. Enter the following command:

   `dm::model refresh -` <*package name*> <*data folder*> <*model name*> <*model generation scenario file*>

   Where:
4. <*package name*> is the name of the Connector package to refresh. Valid options are:
   - Base
   - DemandPointHistory
   - TimeSeries
5. <*data folder*> is the path to the folder containing the EnterpriseOne Supply Chain Business Modeler data that will be the basis for the demand model.
6. <*model name*> is the name for the demand model.
7. <*model generation scenario file*> is the model generation file created by the EnterpriseOne Supply Chain Business Modeler.

Using the Publish Command

This section provides an overview of the publish command and discusses how to use the publish command.

Overview of the Publish Command

When you run the publish command, the system transfers a forecast from Demand Management into XML or gzip format, the latter being a compressed xml format. This forecast can then be imported into the EnterpriseOne Supply Chain Business Modeler.

To use the publish command:

1. At the command prompt, navigate to the start directory (for example, *c:/scp/9.0/common/start)* where the application is installed.
2. Type `run_dm_dm_connector` and then press Enter.
3. Enter the following command:

   `dm::model publish -` *<data folder> <demand model name> <forecast version><format>*

   Where:

   - *<data folder>* is the path to the SCBM Import extract area.
   - *<demand model name>* is the name for the demand model to be published.
   - *<forecast version>* is the name of the forecast version to be published
   - *<format>* is the format of the output file. Options are –xml and –gz.
   - *<rDef file>* is the model generation scenario in SCBM that contains the valid Demand Management parameters required to communicate with SCBM. The file being referenced is *scenarioName*.mgr, where *scenarioName* represents the model generation scenario in SCBM.

Glossary of JD Edwards EnterpriseOne Terms

Accessor Methods/Assessors Java methods to "get" and "set" the elements of a value object or other source file.

activity rule The criteria by which an object progresses from one given point to the next in a flow.

add mode A condition of a form that enables users to input data.

Advanced Planning Agent (APAg) A JD Edwards EnterpriseOne tool that can be used to extract, transform, and load enterprise data. APAg supports access to data sources in the form of rational databases, flat file format, and other data or message encoding, such as XML.

alternate currency A currency that is different from the domestic currency (when dealing with a domestic-only transaction) or the domestic and foreign currency of a transaction.

In JD Edwards EnterpriseOne Financial Management, alternate currency processing enables you to enter receipts and payments in a currency other than the one in which they were issued.

Application Server Software that provides the business logic for an application program in a distributed environment. The servers can be Oracle Application Server (OAS) or WebSphere Application Server (WAS).

as if processing A process that enables you to view currency amounts as if they were entered in a currency different from the domestic and foreign currency of the transaction.

as of processing A process that is run as of a specific point in time to summarize transactions up to that date. For example, you can run various JD Edwards EnterpriseOne reports as of a specific date to determine balances and amounts of accounts, units, and so on as of that date.

Auto Commit Transaction A database connection through which all database operations are immediately written to the database.

back-to-back process A process in JD Edwards EnterpriseOne Supply Management that contains the same keys that are used in another process.

batch processing A process of transferring records from a third-party system to JD Edwards EnterpriseOne.

In JD Edwards EnterpriseOne Financial Management, batch processing enables you to transfer invoices and vouchers that are entered in a system other than JD Edwards EnterpriseOne to JD Edwards EnterpriseOne Accounts Receivable and JD Edwards EnterpriseOne Accounts Payable, respectively. In addition, you can transfer address book information, including customer and supplier records, to JD Edwards EnterpriseOne.

batch server A server that is designated for running batch processing requests. A batch server typically does not contain a database nor does it run interactive applications.

batch-of-one immediate A transaction method that enables a client application to perform work on a client workstation, then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks.

See also direct connect and store-and-forward.

best practices Non-mandatory guidelines that help the developer make better design decisions.

BPEL Abbreviation for *Business Process Execution Language*, a standard web services orchestration language, which enables you to assemble discrete services into an end-to-end process flow.

BPEL PM Abbreviation for *Business Process Execution Language Process Manager*, a comprehensive infrastructure for creating, deploying, and managing BPEL business processes.

Build Configuration File Configurable settings in a text file that are used by a build program to generate ANT scripts. ANT is a software tool used for automating build processes. These scripts build published business services.

build engineer An actor that is responsible for building, mastering, and packaging artifacts. Some build engineers are responsible for building application artifacts, and some are responsible for building foundation artifacts.

Build Program A WIN32 executable that reads build configuration files and generates an ANT script for building published business services.

business analyst An actor that determines if and why an EnterpriseOne business service needs to be developed.

business function A named set of user-created, reusable business rules and logs that can be called through event rules. Business functions can run a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the application programming interfaces (APIs) that enable them to be called from a form, a database trigger, or a non-JD Edwards EnterpriseOne application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.

business function event rule See named event rule (NER).

business service EnterpriseOne business logic written in Java. A business service is a collection of one or more artifacts. Unless specified otherwise, a business service implies both a published business service and business service.

business service artifacts Source files, descriptors, and so on that are managed for business service development and are needed for the business service build process.

business service class method A method that accesses resources provided by the business service framework.

business service configuration files Configuration files include, but are not limited to, interop.ini, JDBj.ini, and jdelog.properties.

business service cross reference A key and value data pair used during orchestration. Collectively refers to both the code and the key cross reference in the WSG/XPI based system.

business service cross-reference utilities Utility services installed in a BPEL/ESB environment that are used to access JD Edwards EnterpriseOne orchestration cross-reference data.

business service development environment A framework needed by an integration developer to develop and manage business services.

business services development tool Otherwise known as JDeveloper.

business service EnterpriseOne object A collection of artifacts managed by EnterpriseOne LCM tools. Named and represented within EnterpriseOne LCM similarly to other EnterpriseOne objects like tables, views, forms, and so on.

business service framework Parts of the business service foundation that are specifically for supporting business service development.

business service payload An object that is passed between an enterprise server and a business services server. The business service payload contains the input to the business service when passed to the business services server. The business service payload contains the results from the business service when passed to the Enterprise Server. In the case of notifications, the return business service payload contains the acknowledgement.

business service property Key value data pairs used to control the behavior or functionality of business services.

Business Service Property Admin Tool An EnterpriseOne application for developers and administrators to manage business service property records.

business service property business service group A classification for business service property at the business service level. This is generally a business service name. A business service level contains one or more business service property groups. Each business service property group may contain zero or more business service property records.

business service property categorization A way to categorize business service properties. These properties are categorized by business service.

business service property key A unique name that identifies the business service property globally in the system.

business service property utilities A utility API used in business service development to access EnterpriseOne business service property data.

business service property value A value for a business service property.

business service repository A source management system, for example ClearCase, where business service artifacts and build files are stored. Or, a physical directory in network.

business services server The physical machine where the business services are located. Business services are run on an application server instance.

business services source file or business service class One type of business service artifact. A text file with the .java file type written to be compiled by a Java compiler.

business service value object template The structural representation of a business service value object used in a C-business function.

Business Service Value Object Template Utility A utility used to create a business service value object template from a business service value object.

business services server artifact The object to be deployed to the business services server.

business view A means for selecting specific columns from one or more JD Edwards EnterpriseOne application tables whose data is used in an application or report. A business view does not select specific rows, nor does it contain any actual data. It is strictly a view through which you can manipulate data.

central objects merge A process that blends a customer's modifications to the objects in a current release with objects in a new release.

central server A server that has been designated to contain the originally installed version of the software (central objects) for deployment to client computers. In a typical JD Edwards EnterpriseOne installation, the software is loaded on to one machine—the central server. Then, copies of the software are pushed out or downloaded to various workstations attached to it. That way, if the software is altered or corrupted through its use on workstations, an original set of objects (central objects) is always available on the central server.

charts Tables of information in JD Edwards EnterpriseOne that appear on forms in the software.

check-in repository A repository for developers to check in and check out business service artifacts. There are multiple check-in repositories. Each can be used for a different purpose (for example, development, production, testing, and so on).

connector Component-based interoperability model that enables third-party applications and JD Edwards EnterpriseOne to share logic and data. The JD Edwards EnterpriseOne connector architecture includes Java and COM connectors.

contra/clearing account A general ledger account in JD Edwards EnterpriseOne Financial Management that is used by the system to offset (balance) journal entries. For example, you can use a contra/clearing account to balance the entries created by allocations in JD Edwards EnterpriseOne Financial Management.

Control Table Workbench An application that, during the Installation Workbench processing, runs the batch applications for the planned merges that update the data dictionary, user-defined codes, menus, and user override tables.

control tables merge A process that blends a customer's modifications to the control tables with the data that accompanies a new release.

correlation data The data used to tie HTTP responses with requests that consist of business service name and method.

cost assignment The process in JD Edwards EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects.

cost component In JD Edwards EnterpriseOne Manufacturing, an element of an item's cost (for example, material, labor, or overhead).

credentials A valid set of JD Edwards EnterpriseOne username/password/environment/role, EnterpriseOne session, or EnterpriseOne token.

cross-reference utility services Utility services installed in a BPEL/ESB environment that access EnterpriseOne cross-reference data.

cross segment edit A logic statement that establishes the relationship between configured item segments. Cross segment edits are used to prevent ordering of configurations that cannot be produced.

currency restatement The process of converting amounts from one currency into another currency, generally for reporting purposes. You can use the currency restatement process, for example, when many currencies must be restated into a single currency for consolidated reporting.

cXML A protocol used to facilitate communication between business documents and procurement applications, and between e-commerce hubs and suppliers.

database credentials A valid database username/password.

database server A server in a local area network that maintains a database and performs searches for client computers.

Data Source Workbench An application that, during the Installation Workbench process, copies all data sources that are defined in the installation plan from the Data Source Master and Table and Data Source Sizing tables in the Planner data source to the system-release number data source. It also updates the Data Source Plan detail record to reflect completion.

date pattern A calendar that represents the beginning date for the fiscal year and the ending date for each period in that year in standard and 52-period accounting.

denominated-in currency The company currency in which financial reports are based.

deployment artifacts Artifacts that are needed for the deployment process, such as servers, ports, and such.

deployment server A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations.

detail information Information that relates to individual lines in JD Edwards EnterpriseOne transactions (for example, voucher pay items and sales order detail lines).

direct connect A transaction method in which a client application communicates interactively and directly with a server application.

See also batch-of-one immediate and store-and-forward.

Do Not Translate (DNT) A type of data source that must exist on the iSeries because of BLOB restrictions.

dual pricing The process of providing prices for goods and services in two currencies.

duplicate published business services authorization records Two published business services authorization records with the same user identification information and published business services identification information.

embedded application server instance An OC4J instance started by and running wholly within JDeveloper.

edit code A code that indicates how a specific value for a report or a form should appear or be formatted. The default edit codes that pertain to reporting require particular attention because they account for a substantial amount of information.

edit mode A condition of a form that enables users to change data.

edit rule A method used for formatting and validating user entries against a predefined rule or set of rules.

Electronic Data Interchange (EDI) An interoperability model that enables paperless computer-to-computer exchange of business transactions between JD Edwards EnterpriseOne and third-party systems. Companies that use EDI must have translator software to convert data from the EDI standard format to the formats of their computer systems.

embedded event rule An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with the business function event rule.

Employee Work Center A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages.

enterprise server A server that contains the database and the logic for JD Edwards EnterpriseOne.

Enterprise Service Bus (ESB) Middleware infrastructure products or technologies based on web services standards that enable a service-oriented architecture using an event-driven and XML-based messaging framework (the bus).

EnterpriseOne administrator An actor responsible for the EnterpriseOne administration system.

EnterpriseOne credentials A user ID, password, environment, and role used to validate a user of EnterpriseOne.

EnterpriseOne object A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects.

EnterpriseOne development client Historically called "fat client," a collection of installed EnterpriseOne components required to develop EnterpriseOne artifacts, including the Microsoft Windows client and design tools.

EnterpriseOne extension A JDeveloper component (plug-in) specific to EnterpriseOne. A JDeveloper wizard is a specific example of an extension.

EnterpriseOne process A software process that enables JD Edwards EnterpriseOne clients and servers to handle processing requests and run transactions. A client runs one process, and servers can have multiple instances of a process. JD Edwards EnterpriseOne processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes don't have to wait if the server is particularly busy.

EnterpriseOne resource Any EnterpriseOne table, metadata, business function, dictionary information, or other information restricted to authorized users.

Environment Workbench An application that, during the Installation Workbench process, copies the environment information and Object Configuration Manager tables for each environment from the Planner data source to the system-release number data source. It also updates the Environment Plan detail record to reflect completion.

escalation monitor A batch process that monitors pending requests or activities and restarts or forwards them to the next step or user after they have been inactive for a specified amount of time.

event rule A logic statement that instructs the system to perform one or more operations based on an activity that can occur in a specific application, such as entering a form or exiting a field.

explicit transaction Transaction used by a business service developer to explicitly control the type (auto or manual) and the scope of transaction boundaries within a business service.

exposed method or value object Published business service source files or parts of published business service source files that are part of the published interface. These are part of the contract with the customer.

facility An entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. A facility is sometimes referred to as a "business unit."

fast path A command prompt that enables the user to move quickly among menus and applications by using specific commands.

file server A server that stores files to be accessed by other computers on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files, but also manages them and maintains order as network users request files and make changes to these files.

final mode The report processing mode of a processing mode of a program that updates or creates data records.

foundation A framework that must be accessible for execution of business services at runtime. This includes, but is not limited to, the Java Connector and JDBj.

FTP server A server that responds to requests for files via file transfer protocol.

header information Information at the beginning of a table or form. Header information is used to identify or provide control information for the group of records that follows.

HTTP Adapter A generic set of services that are used to do the basic HTTP operations, such as GET, POST, PUT, DELETE, TRACE, HEAD, and OPTIONS with the provided URL.

instantiate A Java term meaning "to create." When a class is instantiated, a new instance is created.

integration developer The user of the system who develops, runs, and debugs the EnterpriseOne business services. The integration developer uses the EnterpriseOne business services to develop these components.

integration point (IP) The business logic in previous implementations of EnterpriseOne that exposes a document level interface. This type of logic used to be called XBPs. In EnterpriseOne 8.11, IPs are implemented in Web Services Gateway powered by webMethods.

integration server A server that facilitates interaction between diverse operating systems and applications across internal and external networked computer systems.

integrity test A process used to supplement a company's internal balancing procedures by locating and reporting balancing problems and data inconsistencies.

interface table See Z table.

internal method or value object Business service source files or parts of business service source files that are not part of the published interface. These could be private or protected methods. These could be value objects not used in published methods.

interoperability model A method for third-party systems to connect to or access JD Edwards EnterpriseOne.

in-your-face-error In JD Edwards EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.

IServer service This internet server service resides on the web server and is used to speed up delivery of the Java class files from the database to the client.

jargon An alternative data dictionary item description that JD Edwards EnterpriseOne appears based on the product code of the current object.

Java application server A component-based server that resides in the middle-tier of a server-centric architecture. This server provides middleware services for security and state maintenance, along with data access and persistence.

JDBNET A database driver that enables heterogeneous servers to access each other's data.

JDEBASE Database Middleware A JD Edwards EnterpriseOne proprietary database middleware package that provides platform-independent APIs, along with client-to-server access.

JDECallObject An API used by business functions to invoke other business functions.

jde.ini A JD Edwards EnterpriseOne file (or member for iSeries) that provides the runtime settings required for JD Edwards EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running JD Edwards EnterpriseOne. This includes workstations and servers.

JDEIPC Communications programming tools used by server code to regulate access to the same data in multiprocess environments, communicate and coordinate between processes, and create new processes.

jde.log The main diagnostic log file of JD Edwards EnterpriseOne. This file is always located in the root directory on the primary drive and contains status and error messages from the startup and operation of JD Edwards EnterpriseOne.

JDENET A JD Edwards EnterpriseOne proprietary communications middleware package. This package is a peer-to-peer, message-based, socket-based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all JD Edwards EnterpriseOne supported platforms.

JDeveloper Project An artifact that JDeveloper uses to categorize and compile source files.

JDeveloper Workspace An artifact that JDeveloper uses to organize project files. It contains one or more project files.

JMS Queue A Java Messaging service queue used for point-to-point messaging.

listener service A listener that listens for XML messages over HTTP.

local repository A developer's local development environment that is used to store business service artifacts.

local standalone BPEL/ESB server A standalone BPEL/ESB server that is not installed within an application server.

Location Workbench An application that, during the Installation Workbench process, copies all locations that are defined in the installation plan from the Location Master table in the Planner data source to the system data source.

logic server A server in a distributed network that provides the business logic for an application program. In a typical configuration, pristine objects are replicated on to the logic server from the central server. The logic server, in conjunction with workstations, actually performs the processing required when JD Edwards EnterpriseOne software runs.

MailMerge Workbench An application that merges Microsoft Word 6.0 (or higher) word-processing documents with JD Edwards EnterpriseOne records to automatically print business documents. You can use MailMerge Workbench to print documents, such as form letters about verification of employment.

Manual Commit transaction A database connection where all database operations delay writing to the database until a call to commit is made.

master business function (MBF) An interactive master file that serves as a central location for adding, changing, and updating information in a database. Master business functions pass information between data entry forms and the appropriate tables. These master functions provide a common set of functions that contain all of the necessary default and editing rules for related programs. MBFs contain logic that ensures the integrity of adding, updating, and deleting information from databases.

master table See published table.

matching document A document associated with an original document to complete or change a transaction. For example, in JD Edwards EnterpriseOne Financial Management, a receipt is the matching document of an invoice, and a payment is the matching document of a voucher.

media storage object Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx.

message center A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user.

messaging adapter An interoperability model that enables third-party systems to connect to JD Edwards EnterpriseOne to exchange information through the use of messaging queues.

messaging server A server that handles messages that are sent for use by other programs using a messaging API. Messaging servers typically employ a middleware program to perform their functions.

Middle-Tier BPEL/ESB Server A BPEL/ESB server that is installed within an application server.

Monitoring Application An EnterpriseOne tool provided for an administrator to get statistical information for various EntepriseOne servers, reset statistics, and set notifications.

named event rule (NER) Encapsulated, reusable business logic created using event rules, rather that C programming. NERs are also called business function event rules. NERs can be reused in multiple places by multiple programs. This modularity lends itself to streamlining, reusability of code, and less work.

nota fiscal In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations.

nota fiscal factura In Brazil, a nota fiscal with invoice information.

See also *nota fiscal.*

Object Configuration Manager (OCM) In JD Edwards EnterpriseOne, the object request broker and control center for the runtime environment. OCM keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, OCM directs access to it using defaults and overrides for a given environment and user.

Object Librarian A repository of all versions, applications, and business functions reusable in building applications. Object Librarian provides check-out and check-in capabilities for developers, and it controls the creation, modification, and use of JD Edwards EnterpriseOne objects. Object Librarian supports multiple environments (such as production and development) and enables objects to be easily moved from one environment to another.

Object Librarian merge A process that blends any modifications to the Object Librarian in a previous release into the Object Librarian in a new release.

Open Data Access (ODA) An interoperability model that enables you to use SQL statements to extract JD Edwards EnterpriseOne data for summarization and report generation.

Output Stream Access (OSA) An interoperability model that enables you to set up an interface for JD Edwards EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing.

package JD Edwards EnterpriseOne objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where on the deployment server the installation program can find them. It is point-in-time snapshot of the central objects on the deployment server.

package build A software application that facilitates the deployment of software changes and new applications to existing users. Additionally, in JD Edwards EnterpriseOne, a package build can be a compiled version of the software. When you upgrade your version of the ERP software, for example, you are said to take a package build.

Consider the following context: “Also, do not transfer business functions into the production path code until you are ready to deploy, because a global build of business functions done during a package build will automatically include the new functions.” The process of creating a package build is often referred to, as it is in this example, simply as “a package build.”

package location The directory structure location for the package and its set of replicated objects. This is usually \\deployment server\release\path_code\package\package name. The subdirectories under this path are where the replicated objects for the package are placed. This is also referred to as where the package is built or stored.

Package Workbench An application that, during the Installation Workbench process, transfers the package information tables from the Planner data source to the system-release number data source. It also updates the Package Plan detail record to reflect completion.

Pathcode Directory The specific portion of the file system on the EnterpriseOne development client where EnterpriseOne development artifacts are stored.

patterns General repeatable solutions to a commonly occurring problem in software design. For business service development, the focus is on the object relationships and interactions. For orchestrations, the focus is on the integration patterns (for example, synchronous and asynchronous request/response, publish, notify, and receive/reply).

planning family A means of grouping end items whose similarity of design and manufacture facilitates being planned in aggregate.

preference profile The ability to define default values for specified fields for a user-defined hierarchy of items, item groups, customers, and customer groups.

print server The interface between a printer and a network that enables network clients to connect to the printer and send their print jobs to it. A print server can be a computer, separate hardware device, or even hardware that resides inside of the printer itself.

pristine environment A JD Edwards EnterpriseOne environment used to test unaltered objects with JD Edwards EnterpriseOne demonstration data or for training classes. You must have this environment so that you can compare pristine objects that you modify.

processing option A data structure that enables users to supply parameters that regulate the running of a batch program or report. For example, you can use processing options to specify default values for certain fields, to determine how information appears or is printed, to specify date ranges, to supply runtime values that regulate program execution, and so on.

production environment A JD Edwards EnterpriseOne environment in which users operate EnterpriseOne software.

production-grade file server A file server that has been quality assurance tested and commercialized and that is usually provided in conjunction with user support services.

Production Published Business Services Web Service Published business services web service deployed to a production application server.

program temporary fix (PTF) A representation of changes to JD Edwards EnterpriseOne software that your organization receives on magnetic tapes or disks.

project In JD Edwards EnterpriseOne, a virtual container for objects being developed in Object Management Workbench.

promotion path The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):

11>21>26>28>38>01

In this path, *11* equals new project pending review, *21* equals programming, *26* equals QA test/review, *28* equals QA test/review complete, *38* equals in production, *01* equals complete. During the normal project promotion cycle, developers check objects out of and into the development path code and then promote them to the prototype path code. The objects are then moved to the productions path code before declaring them complete.

proxy server A server that acts as a barrier between a workstation and the internet so that the enterprise can ensure security, administrative control, and caching service.

published business service EnterpriseOne service level logic and interface. A classification of a published business service indicating the intention to be exposed to external (non-EnterpriseOne) systems.

published business service identification information Information about a published business service used to determine relevant authorization records. Published business services + method name, published business services, or *ALL.

published business service web service — Published business services components packaged as J2EE Web Service (namely, a J2EE EAR file that contains business service classes, business service foundation, configuration files, and web service artifacts).

published table — Also called a master table, this is the central copy to be replicated to other machines. Residing on the publisher machine, the F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

publisher — The server that is responsible for the published table. The F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

pull replication — One of the JD Edwards EnterpriseOne methods for replicating data to individual workstations. Such machines are set up as pull subscribers using JD Edwards EnterpriseOne data replication tools. The only time that pull subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the pull subscriber to the server machine that stores the F98DRPCN table.

QBE — An abbreviation for *query by example*. In JD Edwards EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data.

real-time event — A message triggered from EnterpriseOne application logic that is intended for external systems to consume.

refresh — A function used to modify JD Edwards EnterpriseOne software, or subset of it, such as a table or business data, so that it functions at a new release or cumulative update level, such as B73.2 or B73.2.1.

replication server — A server that is responsible for replicating central objects to client machines.

Rt-Addressing — Unique data identifying a browser session that initiates the business services call request host/port user session.

rules — Mandatory guidelines that are not enforced by tooling, but must be followed in order to accomplish the desired results and to meet specified standards.

quote order — In JD Edwards Procurement and Subcontract Management, a request from a supplier for item and price information from which you can create a purchase order.

In JD Edwards Sales Order Management, item and price information for a customer who has not yet committed to a sales order.

secure by default — A security model that assumes that a user does not have permission to execute an object unless there is a specific record indicating such permissions.

Secure Socket Layer (SSL) — A security protocol that provides communication privacy. SSL enables client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

SEI implementation — A Java class that implements the methods that declare in a Service Endpoint Interface (SEI).

selection — Found on JD Edwards EnterpriseOne menus, a selection represents functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.

serialize — The process of converting an object or data into a format for storage or transmission across a network connection link with the ability to reconstruct the original data or objects when needed.

Server Workbench — An application that, during the Installation Workbench process, copies the server configuration files from the Planner data source to the system-release number

	data source. The application also updates the Server Plan detail record to reflect completion.
Service Endpoint Interface (SEI)	A Java interface that declares the methods that a client can invoke on the service.
SOA	Abbreviation for *Service Oriented Architecture*.
softcoding	A coding technique that enables an administrator to manipulate site-specific variables that affect the execution of a given process.
source repository	A repository for HTTP adapter and listener service development environment artifacts.
spot rate	An exchange rate entered at the transaction level. This rate overrides the exchange rate that is set up between two currencies.
Specification merge	A merge that comprises three merges: Object Librarian merge, Versions List merge, and Central Objects merge. The merges blend customer modifications with data that accompanies a new release.
specification	A complete description of a JD Edwards EnterpriseOne object. Each object has its own specification, or name, which is used to build applications.
Specification Table Merge Workbench	An application that, during the Installation Workbench process, runs the batch applications that update the specification tables.
SSL Certificate	A special message signed by a certificate authority that contains the name of a user and that user's public key in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in the user's public key.
store-and-forward	The mode of processing that enables users who are disconnected from a server to enter transactions and then later connect to the server to upload those transactions.
subscriber table	Table F98DRSUB, which is stored on the publisher server with the F98DRPUB table and identifies all of the subscriber machines for each published table.
superclass	An inheritance concept of the Java language where a class is an instance of something, but is also more specific. "Tree" might be the superclass of "Oak" and "Elm," for example.
supplemental data	Any type of information that is not maintained in a master file. Supplemental data is usually additional information about employees, applicants, requisitions, and jobs (such as an employee's job skills, degrees, or foreign languages spoken). You can track virtually any type of information that your organization needs. For example, in addition to the data in the standard master tables (the Address Book Master, Customer Master, and Supplier Master tables), you can maintain other kinds of data in separate, generic databases. These generic databases enable a standard approach to entering and maintaining supplemental data across JD Edwards EnterpriseOne systems.
table access management (TAM)	The JD Edwards EnterpriseOne component that handles the storage and retrieval of use-defined data. TAM stores information, such as data dictionary definitions; application and report specifications; event rules; table definitions; business function input parameters and library information; and data structure definitions for running applications, reports, and business functions.
Table Conversion Workbench	An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table conversion An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table event rules Logic that is attached to database triggers that runs whenever the action specified by the trigger occurs against the table. Although JD Edwards EnterpriseOne enables event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level.

terminal server A server that enables terminals, microcomputers, and other devices to connect to a network or host computer or to devices attached to that particular computer.

three-tier processing The task of entering, reviewing and approving, and posting batches of transactions in JD Edwards EnterpriseOne.

three-way voucher match In JD Edwards Procurement and Subcontract Management, the process of comparing receipt information to supplier's invoices to create vouchers. In a three-way match, you use the receipt records to create vouchers.

transaction processing (TP) monitor A monitor that controls data transfer between local and remote terminals and the applications that originated them. TP monitors also protect data integrity in the distributed environment and may include programs that validate data and format terminal screens.

transaction processing method A method related to the management of a manual commit transaction boundary (for example, start, commit, rollback, and cancel).

transaction set An electronic business transaction (electronic data interchange standard document) made up of segments.

trigger One of several events specific to data dictionary items. You can attach logic to a data dictionary item that the system processes automatically when the event occurs.

triggering event A specific workflow event that requires special action or has defined consequences or resulting actions.

two-way authentication An authentication mechanism in which both client and server authenticate themselves by providing the SSL certificates to each other.

two-way voucher match In JD Edwards Procurement and Subcontract Management, the process of comparing purchase order detail lines to the suppliers' invoices to create vouchers. You do not record receipt information.

user identification information User ID, role, or *public.

User Overrides merge Adds new user override records into a customer's user override table.

value object A specific type of source file that holds input or output data, much like a data structure passes data. Value objects can be exposed (used in a published business service) or internal, and input or output. They are comprised of simple and complex elements and accessories to those elements.

variance In JD Edwards Capital Asset Management, the difference between revenue generated by a piece of equipment and costs incurred by the equipment.

In JD Edwards EnterpriseOne Project Costing and JD Edwards EnterpriseOne Manufacturing, the difference between two methods of costing the same item (for example, the difference between the frozen standard cost and the current cost is an engineering variance). Frozen standard costs come from the Cost Components table, and the current costs are calculated using the current bill of material, routing, and overhead rates.

versioning a published business service Adding additional functionality/interfaces to the published business services without modifying the existing functionality/interfaces.

Version List merge The Versions List merge preserves any non-XJDE and non-ZJDE version specifications for objects that are valid in the new release, as well as their processing options data.

visual assist Forms that can be invoked from a control via a trigger to assist the user in determining what data belongs in the control.

vocabulary override An alternate description for a data dictionary item that appears on a specific JD Edwards EnterpriseOne form or report.

wchar_t An internal type of a wide character. It is used for writing portable programs for international markets.

web application server A web server that enables web applications to exchange data with the back-end systems and databases used in eBusiness transactions.

web server A server that sends information as requested by a browser, using the TCP/IP set of protocols. A web server can do more than just coordination of requests from browsers; it can do anything a normal server can do, such as house applications or data. Any computer can be turned into a web server by installing server software and connecting the machine to the internet.

Web Service Description Language (WSDL) An XML format for describing network services.

Web Service Inspection Language (WSIL) An XML format for assisting in the inspection of a site for available services and a set of rules for how inspection-related information should be made.

web service proxy foundation Foundation classes for web service proxy that must be included in a business service server artifact for web service consumption on WAS.

web service softcoding record An XML document that contains values that are used to configure a web service proxy. This document identifies the endpoint and conditionally includes security information.

web service softcoding template An XML document that provides the structure for a soft coded record.

Where clause The portion of a database operation that specifies which records the database operation will affect.

Windows terminal server A multiuser server that enables terminals and minimally configured computers to display Windows applications even if they are not capable of running Windows software themselves. All client processing is performed centrally at the Windows terminal server and only display, keystroke, and mouse commands are transmitted over the network to the client terminal device.

wizard A type of JDeveloper extension used to walk the user through a series of steps.

workbench A program that enables users to access a group of related programs from a single entry point. Typically, the programs that you access from a workbench are used to complete a large business process. For example, you use the JD Edwards EnterpriseOne Payroll Cycle Workbench (P07210) to access all of the programs that the system uses to process payroll, print payments, create payroll reports, create journal entries, and update payroll history. Examples of JD Edwards EnterpriseOne workbenches include Service Management Workbench (P90CD020), Line Scheduling Workbench (P3153), Planning Workbench (P13700), Auditor's Workbench (P09E115), and Payroll Cycle Workbench.

work day calendar In JD Edwards EnterpriseOne Manufacturing, a calendar that is used in planning functions that consecutively lists only working days so that component and work order scheduling can be done based on the actual number of work days available. A work

	day calendar is sometimes referred to as planning calendar, manufacturing calendar, or shop floor calendar.
workflow	The automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.
workgroup server	A server that usually contains subsets of data replicated from a master network server. A workgroup server does not perform application or batch processing.
XAPI events	A service that uses system calls to capture JD Edwards EnterpriseOne transactions as they occur and then calls third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested notification when the specified transactions occur to return a response.
XML CallObject	An interoperability capability that enables you to call business functions.
XML Dispatch	An interoperability capability that provides a single point of entry for all XML documents coming into JD Edwards EnterpriseOne for responses.
XML List	An interoperability capability that enables you to request and receive JD Edwards EnterpriseOne database information in chunks.
XML Service	An interoperability capability that enables you to request events from one JD Edwards EnterpriseOne system and receive a response from another JD Edwards EnterpriseOne system.
XML Transaction	An interoperability capability that enables you to use a predefined transaction type to send information to or request information from JD Edwards EnterpriseOne. XML transaction uses interface table functionality.
XML Transaction Service (XTS)	Transforms an XML document that is not in the JD Edwards EnterpriseOne format into an XML document that can be processed by JD Edwards EnterpriseOne. XTS then transforms the response back to the request originator XML format.
Z event	A service that uses interface table functionality to capture JD Edwards EnterpriseOne transactions and provide notification to third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested to be notified when certain transactions occur.
Z table	A working table where non-JD Edwards EnterpriseOne information can be stored and then processed into JD Edwards EnterpriseOne. Z tables also can be used to retrieve JD Edwards EnterpriseOne data. Z tables are also known as interface tables.
Z transaction	Third-party data that is properly formatted in interface tables for updating to the JD Edwards EnterpriseOne database.

Index

A

B

E

F

R

S