ORACLE

JD EDWARDS ENTERPRISEONE

EnterpriseOne Supply Chain Planning Advanced Planning Agent 8.12.1 Implementation Guide

August 2007

ORACLE

EnterpriseOne Supply Chain Planning Advanced Planning Agent 8.12.1 Implementation Guide
SKU E1SCP8121SAP-B 0807

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 PeopleSoft 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

Copyright (c) 1999 Wolfram Gloger

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

Copyright (c) 1990, 1993, 1994 The Regents of the University of California. All rights reserved.

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)

This software is copyrighted by the Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, ActiveState Corporation and other parties. The following terms apply to all files associated with the software unless explicitly disclaimed in individual files.

The authors hereby grant permission to use, copy, modify, distribute, and license this software and its documentation for any purpose, provided that existing copyright notices are retained in all copies and that this notice is included verbatim in any distributions. No written agreement, license, or royalty fee is required for any of the authorized uses. Modifications to this software may be copyrighted by their authors and need not follow the licensing terms described here, provided that the new terms are clearly indicated on the first page of each file where they apply.

IN NO EVENT SHALL THE AUTHORS OR DISTRIBUTORS BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE, ITS DOCUMENTATION, OR ANY DERIVATIVES THEREOF, EVEN IF THE AUTHORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE AUTHORS AND DISTRIBUTORS SPECIFICALLY DISCLAIM ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. THIS SOFTWARE IS PROVIDED ON AN "AS IS" BASIS, AND THE AUTHORS AND DISTRIBUTORS HAVE NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

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 (c) 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 (c) 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 (c) 1991, 1992 Bell Communications Research, Inc. (Bellcore)

Copyright (c) 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.

Table of Contents

Preface xi

Related Documentation ... xi
Obtaining Documentation Updates ... xi
Ordering Printed Documentation ... xi

Typographical Conventions and Visual Cues ... xii
Typographical Conventions ... xii
Visual Cues ... xii
Comments and Suggestions ... xiii

Getting Started with Advanced Planning Agent 1

Advanced Planning Agent Overview ... 1

Advanced Planning Agent Business Processes ... 1

Advanced Planning Agent Integrations ... 1

Advanced Planning Agent Implementation ... 2

Understanding Advanced Planning Agent 5

Integrating Systems Using Advanced Planning Agent ... 5

Advanced Planning Agent Components ... 6

Understanding Implementing Advanced Planning Agent ... 7
Information Requirements ... 8
Expertise Required ... 8
Planning Steps ... 9

Managing Advanced Planning Agent Databases and Users 11

Starting the Advanced Planning Agent Administrator ... 11
Starting the Advanced Planning Agent Administrator in Windows ... 11
Starting the Advanced Planning Agent Administrator from a Command Line in Windows ... 12
Starting the Advanced Planning Agent Administrator in UNIX ... 12

Creating System Repositories ... 13
Understanding System Repositories ... 13
Windows Used to View, Select, and Create System Repositories ... 15
Creating Custom System Repositories ... 16

Updating Strategic Network Optimization and Production Scheduling - Process Versions in Configuration Files ... 17
Understanding Configuration Files ... 17
Windows Used to Maintain Configuration Files ... 20
Updating Strategic Network Optimization and Production Scheduling - Process Versions in Configuration Files ... 21

Creating and Managing Storage Definitions ... 22

Understanding Storage Definitions 22
Windows Used to Create and Manage Storage Definitions 22
Creating Storage Definitions 22

Creating and Managing Data Object Definitions 24
Understanding Data Object Definitions 24
Windows Used to Create and Manage Data Object Definitions 25
Creating Data Object Definitions 26
Creating Data Object Definitions Based on a Database 26
Saving Data Object Definitions 27
Deleting Data Object Definitions 27
Printing Data Object Definitions 28
Edit Data Object Field Definitions 28
Editing Data Object Field Definition Values 29

Connecting to Databases 32
Understanding Databases 33
Establishing Relational Properties Using Foreign Keys 35
Mapping Data Types 36
Converting Float Data Types to Double Data Types 36
Connecting to Relational Databases 37
Interfacing with DB2 37
Interfacing with Oracle 38
Interfacing with Microsoft SQL Server 40

Managing Advanced Planning Agent Users 42
Understanding Read and Write Permissions 42
Windows Used to Manage Advanced Planning Agent Users 42
Managing User Access Rights 43
Setting Browse Permissions 44

Creating and Managing Advanced Planning Agent Projects 45

Starting the Advanced Planning Agent Manager 45
Starting the Advanced Planning Agent Manager in Windows 45
Starting the Advanced Planning Agent Manager from a Command Line in Windows 46
Starting the Advanced Planning Agent Manager in UNIX 46

Creating and Opening Projects 47
Windows Used to Create and Open Projects 47
Refreshing Project Path Names in the Data Flow Editor 47
Matching Patterns Using Wildcard Characters 48

Specifying Storage Connections and Data Objects 48
Understanding Storage Connections and Data Objects 48
Windows Used to Create and Manage Storage Connections and Data Objects 49
Creating Storage Connections 50
Setting Parameters for Storage Connections 50

Creating and Managing Data Flows 51

Understanding Data Flows 51
Nodes 51
Arcs and Attach Points 53
Subflows 54
Verifying Data Flows 57
Viewing Different Areas of the Data Flow Workspace 57

Creating and Managing Data Flows 58

Windows Used to Create and Manage Data Flows 58
Creating Nodes in Data Flows 59
Moving Nodes in Data Flows 59
Aligning Nodes in Data Flows 59
Connecting Nodes with Arcs 59
Adding Subflows to Data Flows 60

Setting up Data Object Nodes **61**

Understanding Data Object Node Options 61

Understanding Handling Exception Output from Data Object Nodes 66

Understanding Viewing and Editing Data in Data Object Nodes 67
Windows Used to Set up Data Object Nodes 67

Viewing and Editing Selection Criteria Fields and Functions 68

Setting up Selection Filter Nodes **69**

Understanding Selection Filter Nodes 69

Understanding the Union Feature in Selection Filter Nodes 70

Understanding Selecting Data from Input Data Objects 71
Windows Used to Set up Selection Filter Nodes 74

Setting up Selection Filter Nodes 75

Selecting Fields and Functions 75

Setting up Aggregation Filter Nodes **77**

Understanding Aggregation Filter Nodes 77

Determining Data that is Selected from Input Data Objects 77

Determining Data that is Grouped and Sorted 78
Grouping Information 78
Performing a Rolling Function Sort in the Aggregation Filter 78
Using Reset Sort Keys 79
Using Rolling Sort Keys 79

Determining Aggregated Data that is Selected 80

Determining Data that is Output 80
Example: Using an Aggregation Filter 80
Example: Aggregation Filters Doing Rolling Function Sorts 82

Bucketizing Data in Aggregation Filter Nodes 84
Bucketvalues Method 84
Bucketlengths Method 84
Bucketperiods Method 85
Bucketizing Examples 85
Handling Gaps and Overlaps in Horizons When You Bucketize 87

Windows Used to Set up Aggregation Filter Nodes 89

Setting up Aggregation Filter Nodes 90

Setting up Join Filter Nodes **93**

Understanding Join Filter Nodes 93

Join Filter Nodes 93
Determining Data that is Selected From Input Data Objects 93
Determining Data that is Joined 94
Determining Joined Data that is Selected in Join Filter Nodes 94
Determining Data that is Output 94
Collecting Data that is not Joined in Join Filter Nodes 94
Window Used to Set up Join Filter Nodes 99
Setting up Join Filter Nodes 99

Setting up Production Scheduling - Process Nodes 101

Understanding Production Scheduling - Process Nodes 101
Window Used to Set up Production Scheduling - Process Nodes 102
Setting up Production Scheduling - Process Nodes 102

Creating and Managing Strategic Network Optimization Nodes 105

Understanding Strategic Network Optimization Nodes 105
Strategic Network Optimization Nodes 105
Specifying Input and Output Files 106
Using Strategic Network Optimization Nodes Without Launching the System 107
Window Used to Set up Strategic Network Optimization Nodes 107
Setting up Strategic Network Optimization Nodes 107

Creating and Managing Data Editor Nodes 109

Understanding Data Editor Nodes 109
Data Editor Nodes 109
Displaying and Printing Data When you Run Data Flows 109
How Selection Criteria in Data Object Nodes Affect the Data Editor 110
Windows Used to Set up Data Editor Nodes 111
Setting up Data Editor Nodes 111

Setting up Shell Nodes 113

Understanding Shell Nodes 113
Shell Nodes 113
Using Parameters in Shell Nodes 113
Window Used to Set up Shell Nodes 114
Setting up Shell Nodes 114

Running Data Flows 117

Understanding Running Data Flows 117
Windows Used to Run Data Flows 121
Running Data Flows from the Project Window 121
Running Data Flows From the Data Flow Launcher 121
Running Data Flows in Batch 122

Creating Batch Files for Running Data Flows 122
Running Batch Files 123
Running Data Flows in Batch Using Parameters 123
Specifying the Repository and Resource Area When Running Data Flows in Batch 124
Specifying the Advanced Planning Agent Version Number Before Running Batch Files 124
Running Batch Files after Specifying the Advanced Planning Agent Version 124
Passing Environment Variables to Scripts 124
Verifying that Data Flows Have Run in Batch 125
Viewing and Managing Error Messages 126
Understanding Error Messages 126
Viewing Error Messages After Running Data Flows 126
Clearing Error Indicators 127
Viewing Error Messages 127
Specifying Errors that You Want to View 128
Compressing Identical Errors Into One Message 128
Highlighting Errors 128
Viewing Errors Individually 129
Refreshing Error Data 129
Organizing Log Output 129
Understanding Managing Logs 129
Window Used to Manage Logs 130
Changing the Location for Log Output 130
Changing the Maximum Log Size 130
Changing the Truncate Value of Logs 130
Viewing and Reporting Data When You Run Data Flows 131
Displaying the Data Editor, Production Scheduling - Process, or Strategic Network Optimization 131
Refreshing the Data Flow Editor 131
Printing Reports from the Data Editor 131
Clearing Run Information 131
Using Arcs to Map Data Types 132
Using Parameters with Data Flows 132
Adding Parameters at the Project Level 133
Adding Parameters at the Data Flow Level 133
Editing Parameter Names 133
Editing Parameter Default Values 133
Deleting Parameters or Default Values 134
Overriding Default Parameters when you Run Data Flows 134
Specifying Parameters In a Properties Window 134
Specifying Parameters In a Selection Criteria Window 135
Monitoring Systems for Space 135
Using Tool Command Language 138
Understanding Tool Command Language 138
Creating Tcl Scripts in Data Flows 138
Troubleshooting Data Flows 139

Viewing and Editing Data 141

Customizing Views 141
Moving Rows and Columns 142
Swapping Rows and Columns 142
Hiding and Revealing Row and Column Labels 142
Showing Data in Layers 142
Moving Data from Layers 143
Displaying Headings 143

Working with Data 143
Adding Records 144
Adding Rows 144
Adding Columns 145
Deleting Records 145
Deleting Records When Data Tables Have Only Primary Keys 145
Deleting Rows 145
Deleting Columns 146
Changing the Date Format 146
Defining the Number of Decimal Places 147
Changing Values in Rows, Columns, or Views 147

Selecting Data Classifications and Types 148
Creating a Period Durations File 148

Using Breaks 149
Performing Calculations with Breaks 149
Showing or Hiding Breaks 150

Using Expressions 150
Creating Fields and Expressions 150
Editing Expressions 151
Deleting Expressions 151
Recalculating Expressions 151

Assigning Values to Fields 152
Recalculating Inverse Formulas 152
Using Expressions from Different Views 153
Highlighting Expressions and Fields 153
Copying Expressions 155
Highlighting Specific Fields 155
Highlighting Expressions from Different Views 156
Viewing Available Functions 156
Using Arithmetic Symbols 156
Configuring Expressions in Data Editor Functions 157
Copying Expressions 158

Saving Views 158
Loading Views 159
Deleting Views 160
Printing Reports 160

Saving to Database Tables 162

Exporting Data to Files 162

Using the Inverse Formula to Automatically Update Data 163

Understanding Expressions and Functions 164

Advanced Planning Agent and Data Editor Functions 164

Date Functions 171
Color Functions 172
Trigonometric Functions 173
Rolling Functions 174
String Functions 174
Function Reference 175
ABS 179
ACOS 180
AND, OR, XOR Comparisons 180
ABSDATE 182
ADVANCEDAYS 183
ASCIICONVERT 184
ASIN 185
ASSIGN 185
ATAN 186
ATAN2 187
AVG 187
BACKGROUND 188
BETWEEN 188
BLACK 189
BLUE 189
BUCKETLENGTHS 190
BUCKETPERIODS 190
BUCKETVALUES 191
COS 191
COSH 192
COVER 192
Using the #timeUnit Parameter 193
DATETOUMTSEC 195
DAY 196
DAYDUR 197
Depletion 198
EQTOL 200
ERROR 200
EXP 201
FALSE 201
FINDNEXTDAY 201
GAMMA 202
GETDATE 202
GETDAYSINMONTH 203
GETENV 204
GETOL 205
GREEN 205
GTTOL 206
HOUR 206
HOURDUR 207
IF 209
INDEX 209
INFINITESIMAL 210
INFINITY 210

INSERT 211
INT 212
InverseDepletion 213
ISNULL 214
KeepBlanks 214
KillBlanks 215
LETOL 215
LN 216
LOG 216
LTRIMBLANKS 216
LTTOL 217
MAKEDATE 218
MAKEDATEFROMWEEK 219
MAKESYSKEY 222
MAX 222
MIN 223
MINUTE 223
MINUTEDUR 224
MONTH 226
NEG 227
NETOL 227
NEWSYSKEY 228
Next 228
NEXTMONTH 229
NEXTYEAR 231
NOT 231
NOW 231
NULL 232
NULLDEF 233
ORANGE 233
PERCENT 233
OVERUNDER 234
PARSESTRING 234
PI 235
PINK 235
POS 236
POWER 236
PREV 236
PRODUCT 237
R_AVG 238
R_COVER 238
R_MAX 241
R_MIN 242
R _PREV 242
R_PRODUCT 244
R_SUM 244
RAND 245
RED 245
REGEXPR 245
See Also 246
RelOffset 246
REMAINDER 248
REMOVE 249
REPLACEINDEX 250
REPLACEPATTERN 250
ROUND 251
RTRIMBLANKS 252

SECOND 253
SECONDDUR 254
SEQUENCE 255
SIGN 255
SIN 256
SINH 256
SQRT 257
STDDEV 257
STRINGCAST 257
STRINGCONCAT 258
STRINGLENGTH 259
STRINGTODOUBLE 260
STRINGTOINT 261
SUBSTRING 262
SUM 263
TAN 263
TANH 263
TOLOWER 264
TOUPPER 265
TRIMBLANKS 266
TRUE 267
VIOLET 267
WEEKDAY 268
WEEKDUR 269
WEEKNUMBER 270
YEAR 271
YEARDAYNUMBER 272
YELLOW 273
ZERO 274

Packing Project Files **275**

Starting the Pack Utility 275
Starting the Pack Utility in Windows 275
Starting the Pack Utility in UNIX 275
Packing Projects 276
Unpacking a Project 277
Exporting Data Using the Pack Utility 277
Importing Data Using the Pack Utility 278

XML Support **281**

Understanding Advanced Planning Agent Support for XML 281
Transforming Relational Data Tables into XML Documents 282
Running the xmlencode Program 282
Using Run-time Parameters 284
Controlling Which Configuration File Parameters Appear in Output Files 284
Controlling Element Naming 285
Specifying the Output Style 285
Creating Input Relational Data Files 288
Creating Configuration Files 288

Transforming XML Documents into Relational Data Tables 293
Running the xmlparse Program 294
Creating xmlparse Configuration Files 294
Specifying the S2 Standard XML Schema 297
Specify Supply Chain Planning Encoding XML 300

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 other Implementation Guide.

Related Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on the Customer Connection web site. Through the Documentation section of Customer Connection, you can download files to add to your documentation library. You'll find a variety of useful and timely materials, including updates to the full documentation that is delivered on the CD-ROM.

Important!

Before you upgrade, you must check Customer Connection for updates to the upgrade instructions. EnterpriseOne Supply Chain Planning continually posts updates as the upgrade process is refined.

See Also

Customer Connection web site, http://www.peoplesoft.com/corp/en/login.asp

Ordering Printed Documentation

You can order printed, bound volumes of the complete documentation that is delivered on your CD-ROM. Oracle EnterpriseOne makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed documentation by using any of these methods:

- Web

- Telephone

Web

From the Documentation section of the Customer Connection web site, access the PeopleSoft Press web site under the Ordering PeopleBooks topic. The PeopleSoft Press web site is a joint venture between PeopleSoft and Consolidated Publications Incorporated (CPI), the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

Telephone

Contact CPI at 800 888 3559.

See Also

PeopleSoft Customer Connection web site, http://www.peoplesoft.com/corp/en/login.asp

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.

Typographical Conventions

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

Typographical Convention or Visual Cue	Description
Italics	Indicates field values, emphasis, and Implementation guide or other book-length publication titles. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the number *0*, not the letter *O*.
“ ” (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
Cross-references	Implementation guides provide cross-references either below the heading “See Also” or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Visual Cues

Implementation guides contain the following visual cues.

Notes

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

Important!

Example of an important note.

Warnings

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

Warning!

Example of a warning.

Comments and Suggestions

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

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

Or send email comments to doc@peoplesoft.com.

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

Getting Started with Advanced Planning Agent

This chapter provides an overview of Advanced Planning Agent and discusses:

- Advanced Planning Agent business processes.
- Advanced Planning Agent integrations.
- Advanced Planning Agent implementation.

Advanced Planning Agent Overview

Advanced Planning Agent integrates your systems by linking them together in a single system. You can model your entire supply chain with data flows in Advanced Planning Agent. You can also view and interact with the information in each step in a data flow.

Advanced Planning Agent Business Processes

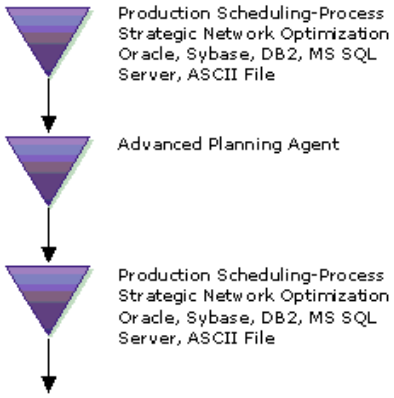
By enabling you to transfer data between supply chain systems, Advanced Planning Agent supports the Production business process. Using Advanced Planning Agent, your company can share data for planning, deploying, producing, and assessing production. Advanced Planning Agent also supports the Order Fulfillment business process by helping organizations share data for capturing, fulfilling, and settling goods sold.

Advanced Planning Agent Integrations

Advanced Planning Agent can integrate data from the following sources:

- JD Edwards EnterpriseOne Strategic Optimization Network.
- JD Edwards EnterpriseOne Production Scheduling - Process.
- Oracle databases.
- DB2 databases.
- Microsoft SQL Server.
- Token-delimited ASCII files.
- Fixed-format ASCII files.

The following diagram illustrates how Advanced Planning Agent converts information between the different data formats of these sources and communicates information between supply chain systems:



Advanced Planning Agent converts data between formats

Advanced Planning Agent Implementation

The Advanced Planning Agent implementation process can be divided into the following steps:

- Installing Advanced Planning Agent and setting up the database environment.
- Creating a system repository and updating configuration files.
- Creating storage and data object definitions.
- Creating and managing user accounts.
- Creating a project file.
- Creating storage connections and data objects.
- Creating nodes and data flows.

Installing Advanced Planning Agent and Creating the Database

Step	Reference
Install Advanced Planning Agent.	Advanced Planning Agent 8.12.1 Installation Guide.
Set up the database environment.	Advanced Planning Agent 8.12.1 Installation Guide.

Creating a System Repository and Updating Configuration Files

Step	Reference
Create a system repository.	Creating System Repositories.
Update configuration files.	Updating Strategic Network Optimization and Production Scheduling - Process Versions in Configuration Files.

Creating Storage and Data Object Definitions

Step	Reference
Create storage definitions.	Creating and Managing Storage Definitions.
Create data object definitions.	Creating and Managing Data Object Definitions.

Creating and Managing User Accounts

Step	Reference
Create and manage user accounts.	Managing Advanced Planning Agent Users.

Creating a Project

Step	Reference
Create a project file.	Creating and Opening Projects.

Creating Storage Connections and Data Objects

Step	Reference
Create storage connections and data objects.	Specifying Storage Connections and Data Objects.

Creating and Running Data Flows

Step	Reference
Create data flows.	Creating and Managing Data Flows.
Run data flows.	Running Data Flows.

Understanding Advanced Planning Agent

This chapter discusses:

- Integrating Systems using Advanced Planning Agent.
- Advanced Planning Agent components.
- Understanding implementing Advanced Planning Agent.

Integrating Systems Using Advanced Planning Agent

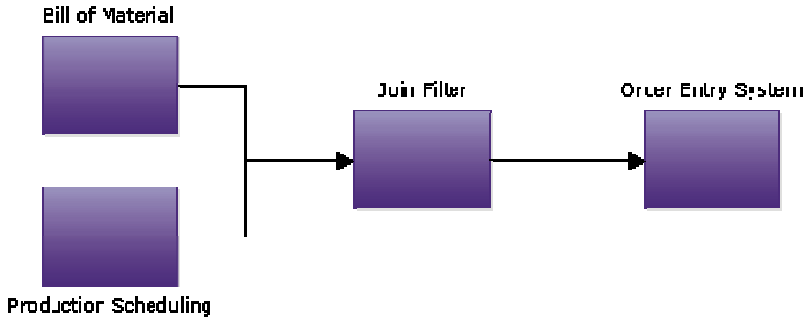
The relationships between systems in a manufacturing environment can be complex. Manufacturing systems are operated by different individuals at different times, often at different locations. Systems might be installed gradually over a long period of time. Some packages are purchased; others are homegrown legacy systems. Systems might share resources and concepts, but they often use different time frames and have different goals. Furthermore, manufacturing environments change rapidly at all levels.

The complexity of manufacturing systems can lead to difficult communications and a complex infrastructure. However, systems must share data in a timely manner to support decision-making. For example, one system might require data for strategic planning at a product family level on a monthly basis. Another system might require data for individual shifts by SKU, for use in a detailed production scheduling package.

To synchronize the data between disparate systems, you can use Advanced Planning Agent. Advanced Planning Agent enables you to flow data between databases and systems and convert data to the format required by specific systems.

After creating system repositories and configuration files that define how Advanced Planning Agent connects to databases and systems, you can create data flows in Advanced Planning Agent Manager. Data flows are used to integrate data from disparate systems into one system. For example, data flows use filters to extract specific information from one system for use in another system.

The following diagram illustrates a simple data flow:



Simple data flow

In this data flow, Advanced Planning Agent joins data from a bill of materials and a schedule that was produced by Production Scheduling-Process. The Join filter in this data flow join the data from these two sources and calculates the net material requirements. This data can then be sent to the database for use by another system, such as an order entry system.

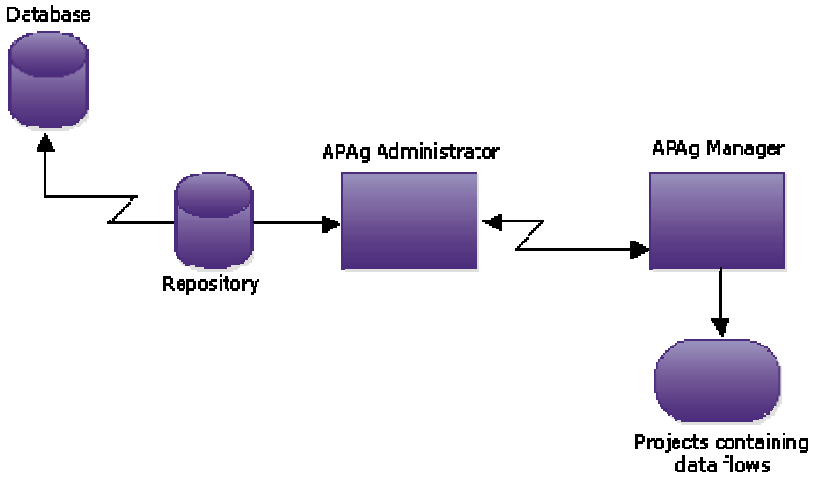
The data flow consists of a series of nodes joined by arcs that represent the flow of information across the collaborative enterprise network. Each node has an icon that represents the type of node and a symbolic name. The overall flow of data is from left to right. When you run a data flow, the node on the left end of an arc runs before the node on the right. If nodes are attached vertically, the flow moves from top to bottom.

Advanced Planning Agent Components

Advanced Planning Agent includes the following three components:

- Advanced Planning Agent Administrator. The Advanced Planning Agent Administrator enables you to create and manage user accounts and database connections. You can use the Administrator to do the following:
  - Manage repositories, which are databases that contain data about your production database.
  - Create storage definitions, which specify how the data flows connect to databases.
  - Create data object definitions, which specify the structure of database objects.
  - Manage user access permissions.
  - Monitor the system for space.
- Advanced Planning Agent Manager. The Advanced Planning Agent Manager enables you to build, test, and maintain data flows. Data flows represent the flow of data across the supply chain. The Advanced Planning Agent Manager includes a data flow editor, where you can build, view, edit, and run data flows.

The following diagram illustrates how the Advanced Planning Agent Manager interacts with the Administrator:



Interaction of Advanced Planning Agent Administrator and Manager

As illustrated in this diagram, a data flow does not read data directly. Instead, the data flow connects to the data indirectly through the Manager, the Administrator, and a repository. The Manager reads projects that contain data flows, the Manager and Administrator communicate, and the Administrator defines the connections to your database.

- Data Flow Launcher. The Data Flow Launcher is similar to the Advanced Planning Agent Manager but does not need the licenses that are required to run flows in the Advanced Planning Agent Manager. Some functions in the Advanced Planning Agent Manager are not available in the Data Flow Launcher. For example, you can run data flows and view the results in the Data Flow Launcher, but you cannot save changes that you make to the data flows. You can use the Data Flow Launcher to:
  - Browse data flows.
  - Browse the data in data object nodes.
  - Run data flows.

Understanding Implementing Advanced Planning Agent

This section describes the background planning and preparation required to build a large-scale Advanced Planning Agent project. This summary is intended as a starting point and a basic guide; it is not comprehensive enough to apply to all situations. This section provides overviews of:

- Information requirements.
- Background planning.
- Planning steps.

Information Requirements

Before you build a large-scale Advanced Planning Agent project, you need the following information:

- Number of systems being introduced and replaced.
- Number of departments involved in the project.
- Resources for technical support.
- Time span of the project.

For each system involved in the project, you should do the following:

- Specify which systems will pass information back and forth.
- Identify the source and destination of all data.
- Translate information flow into data flows.
- Assess how frequently that the data flows will run.
- Determine the size of the data files.
- Links with existing systems.
- Ensure that the requirements of the system are clear.

To support Advanced Planning Agent users, you should determine:

- Who will support the organization.
- Whether geographical location is a problem.
- Whether sufficient information is documented.

Information and knowledge should be documented and distributed to everyone involved in the project.

Expertise Required

To effectively implement an integration project, your project group requires expertise in the following areas:

- Supply chain management systems.
- Multiplant project management.
- Networks, client/server environments, and communications.
- Database design and implementation.
- Systems.
- Project management.

Individuals do not need to be experts in every area, but they should understand the complexities involved, and know where and how to get help when necessary. Depending on your organization, other requirements can exist.

Planning Steps

The following general steps are useful in planning Advanced Planning Agent projects:

1. Determine reporting requirements.
   - Develop a list of all the reports required and wanted.
   - Identify the information sources for the reports.
   - Determine if data should flow one way or two ways.
   - Translate the reports into data flows.
   - Determine if reports are produced in batch or interactively.
2. Decide how the data should be stored.
   - Determine where and how data will be stored (machine location; relational database management system or flat file).
   - Determine whether an existing database can be used and how it is set up.
   - Determine if new tables must be created in the database.
   - Decide who needs access to data.

   After you have determined your systems, report requirements, and data storage needs, draw a flow diagram that indicates where the following are, in relation to the entire model:

   - Data storage
   - Systems (legacy systems, new systems)
   - Reports
3. Design and build the data flows.
   - Design new data tables.
   - Locate and document existing data tables that are to be used.
   - Prioritize the required data flows.
   - Build the data flows in order of priority.
4. Test the data flows for the following:
   - Software bugs
   - Performance
   - Logical errors
5. Define procedures to run the data flows.
   - Train users on how to use Advanced Planning Agent.
   - Identify how to run each flow: the order, in batch or interactively, before or after using a system.
   - Document and simplify the procedures.

Managing Advanced Planning Agent Databases and Users

This chapter discusses how to:

- Start the Advanced Planning Agent Administrator.
- Create system repositories.
- Update Strategic Network Optimization and Production Scheduling-Process versions in configuration files.
- Create and manage storage definitions.
- Create and manage data object definitions.
- Connect to databases.
- Manage Advanced Planning Agent users.

Starting the Advanced Planning Agent Administrator

This section discusses how to:

- Start the Advanced Planning Agent Administrator in Windows.
- Start the Advanced Planning Agent Administrator from a command line in Windows.
- Start the Advanced Planning Agent Administrator in UNIX.

Starting the Advanced Planning Agent Administrator in Windows

To start the Advanced Planning Agent Administrator in Windows:

1. Login with your userid.
2. Select Start, Programs, Supply Chain Planning, Advanced Planning Agent *x.x.x,* APAg Administrator.
3. In the AP Administrator window, enter the password.

Note

The password for the nmx_admin user of the demo repository that is shipped with Advanced Planning Agent is admin. Each repository has its own password for the nmx_admin user. You can specify the password for the nmx_admin user when you create a repository.

4. Click OK.

Note

If you cannot login, ensure that the correct system repository is selected.

See Creating System Repositories.

Starting the Advanced Planning Agent Administrator from a Command Line in Windows

To start the Advanced Planning Agent Administrator from a command line in Windows:

1. Open a command prompt window.
2. Change to the APS/bin directory.
3. Enter the following command:

```
apag [-display hostname:0:0]
```

4. Enter 1.
5. Enter the number for the Advanced Planning Agent version that you want to run.

Starting the Advanced Planning Agent Administrator in UNIX

To start the Advanced Planning Agent Administrator:

1. In a UNIX window, change to the APS/bin directory.
2. Enter the following command:

```
apag
```

3. Type 1 and press Enter.
4. Select the version that you want to use.

   Do not close the UNIX window because Advanced Planning Agent will stop immediately.

5. In the APAg Admin Login window, enter the password.

Note

The password for the nmx_admin user of the demo repository that is shipped with Advanced Planning Agent is admin. Each repository has its own password for the nmx_admin user. You can specify the password for the nmx_admin user when you create a repository.

6. Click OK.

Note

If you cannot login, ensure that the correct system repository is selected.

Creating System Repositories

This section provides an overview of system repositories, lists the windows used to view, select, and create system repositories, and discusses how to create custom system repositories.

Understanding System Repositories

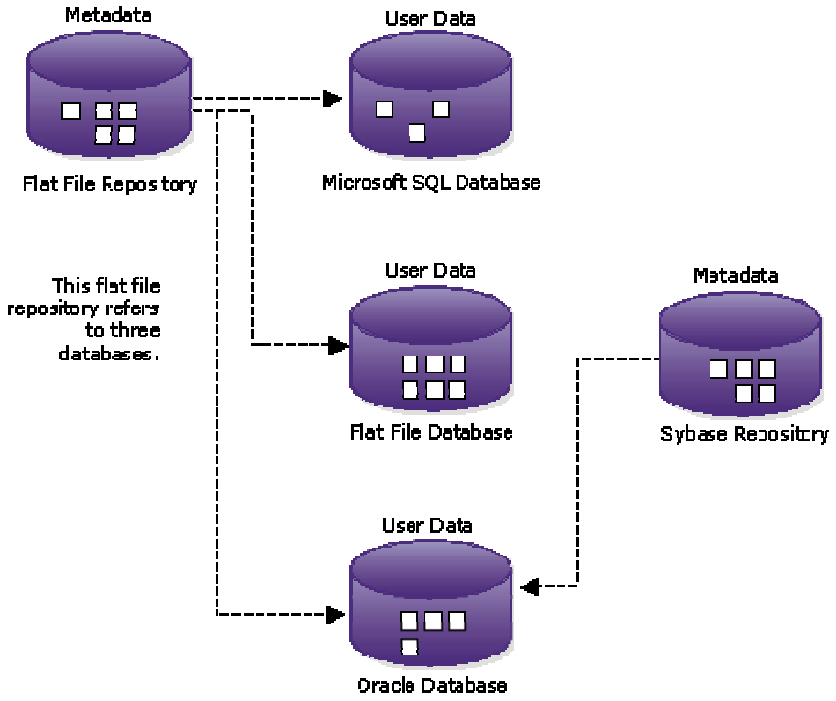
An Advanced Planning Agent system repository is a flat file or Oracle database that contains information about the databases and systems that you want to connect using Advanced Planning Agent. This information, or metadata, includes:

- Names of and information about databases, data tables, and fields in your systems.
- Names of users and their access rights.

You can create multiple system repositories, but you can only use one at a time when you are working in Advanced Planning Agent. The system repository that you use is specified by the value of the NMX_INIT environment variable. When you start Advanced Planning Agent, you can enter a new value for the NMX_INIT environment variable, if desired.

When you create a system repository, you can select whether to create a flat file or Oracle system repository. Each repository type has its benefits. A flat file repository is fast, easy to use, and its contents can be easily edited if necessary. An Oracle repository can offer better backup protection and more security.

The type of repository is not restricted by the production database type. A flat file repository can contain information about an Oracle production database. Similarly, an Oracle repository can contain information about a flat file production database. The following diagram illustrates how a flat file system repository and databases can refer to other databases:



Flat file system repository referring to databases

An Advanced Planning Agent system repository contains the following data tables and information:

Data Table Name	Description
Nmxuser	Contains the names of the users, as displayed in the User Manager, and the users' encrypted passwords.
Nmxstorage	Contains the symbolic names of databases, as displayed in the Storage Definition Manager. This file contains the information about storage definitions. For example, all of the information about your hard delimiters is stored here. It also stores information about relational databases, including type and name.
Nmxaccess	Contains the access rights for each user, as displayed in the Access Rights Manager.
Nmxtable	Contains a list of the data tables, as displayed in the Data Definition Manager.
Nmxcol	Contains a list of the columns and their data types in every data table.

An Advanced Planning Agent repository requires the following files:

File Name	Description
System_key	Contains system keys. You specify the location and name of the system-key file when you create a repository. Multiple repositories can share the system_key file.

File Name	Description
username.cfg	Contains information for each user. Each user has one .cfg file.

Notes

For a flat file system repository, the system_key and username.cfg files can be in the same directory as the data tables. For an Oracle system repository, these files are stored in a separate directory. Do not change these data tables and files.

Metadata and user data can be stored in the same database.

Windows Used to View, Select, and Create System Repositories

Window Name	Navigation	Usage
Environment Info	Select Help, Environment in the Advanced Planning Agent Administrator or Manager window.	View the current system repository name.
UNIX command prompt	Enter the following command at a UNIX command prompt: `echo $NMX_INIT`	View the current system repository name.
Change info	Click Setup in the Advanced Planning Agent Administrator or Manager login window. Enter the system repository name in the NMX_INIT field and click OK.	Select system repositories.
UNIX command prompt	Enter the following command at a UNIX command prompt: `export NMX_INIT=` *system_repository_name* **Note** To select the same system repository each time you login to UNIX, you can add the export command to your *.profile* file.	Select system repositories.
MAKEREP window in Windows	From the Start menu, select Programs, Supply Chain Planning, Advanced Planning Agent *x.x.x,* Create Custom Repository.	Create system repositories in Windows.

Window Name	Navigation	Usage
MAKEREP window in UNIX	Enter the following command at a UNIX prompt: `apag` Select Create Custom Repository from the list of options that appear.	Create system repositories in UNIX.
UNIX command prompt	Enter the following command at a UNIX command prompt: `SCIdir/vers_x.x.x/bin /makeRepBatch` "*<repository_dir>*" "*<syskey_file>*" "*<db_type>*" "*<database_name>*" "*<record_delimiter>*" "*<hard_delimiter>*" "*<soft_delimiter>*" "*<db_user>*" "*<db_password>*" *<sci_admin_password> <repository_name> <resource_area>*	Create system repositories in UNIX.

Creating Custom System Repositories

Access the MAKEREP window.

To create a custom system repository:

Complete the following fields:

Type	Type of repository. Select Oracle only if the Oracle_HOME and Oracle_SID environment variables are defined. The Flatfile Fixed and DB2 file types are not available for hosting an Advanced Planning Agent repository.
NMX_RES	Path to the resource directory. This field is automatically filled in, but you can change it to another existing resource directory.
Name	Name of the repository. The name should be unique and meaningful. This is the NMX_INIT setting.
Directory	Full path and directory name of the new repository. If you specify a directory that already exists, its contents are deleted. You can create a directory only if the path to that directory exists. For example, you can specify the directory /path/example/directory only if /path/example already exists. You can use the Select button to display a window from which you can find and select a directory.
Record	Optional. For flat file repositories only. The end-of-record delimiter. The default value is <newline>.
Hard	Optional. For flat file repositories only. The hard delimiter. The default value is <tab>. When a delimiter is hard, consecutive occurrences count each occurrence as a field separator. For example, consider the following record using <comma> as a hard delimiter: "A",,,,"B" means six fields have the values "A", null, null, null, null, "B".

Soft	Optional. For flat file repositories only. The soft delimiter. The default is to have no soft delimiter. When a delimiter is soft, consecutive occurrences are counted as one field separator. For example, consider the following record using <comma> as a soft delimiter: "A",,, *"B"* means two fields are in this record, which have the values: "A","B".
Database	Optional. For Oracle repositories only. The name of the database.
Connect	Optional. If the DSQUERY variable is set, it is automatically displayed for you.
User ID	Optional. For Oracle repositories only. The userid of a database user with create table permissions.
Password	Optional. For Oracle repositories only. The password of the database user.
System Key File	The full path and file name of the system key file. The system key file can be shared by several repositories. If you specify a file that does not exist, it is created. If you specify a system key file that already exists, it is used as it is and is not overwritten. You can use the Select button to display a window from which you can find and select a system key file.
APAg Admin Password	The password that you want to assign to the Advanced Planning Agent Administrator. This field may be left blank.

Notes

The files that are created in the repository have the same permissions as the user who ran MAKEREP to create the repository. You can change these permissions after the repository is created.

To avoid problems, use the following symbols when using special characters as delimiters: <tab> <newline> <return> <cr> <formfeed> <backspace> <escape> <space> <comma> You are not restricted to these delimiters. Other characters, such ^ or ~, can also be used.

Updating Strategic Network Optimization and Production Scheduling - Process Versions in Configuration Files

This section provides an overview of configuration files, lists the windows used to maintain configuration files, and describes how to update Strategic Network Optimization and Production Scheduling-Process versions in configuration files.

Understanding Configuration Files

When you install Advanced Planning Agent, a resource directory is created. The resource directory contains configuration files that Advanced Planning Agent data flows need to run correctly.

The location of the resource directory is specified by the NMX_RES variable. When you install Advanced Planning Agent, this configuration file directory is created in /path/SCIdir/vers_*x.x.x* /cfg/, where *x.x.x* is the Advanced Planning Agent version number.

When you start Advanced Planning Agent or install a new version of Advanced Planning Agent, the NMX_RES variable is automatically set to the directory of the version that has been run or installed.

If you create a new repository, the repository.cfg file is automatically updated as long as NMX_RES is set. If you create a new repository and NMX_RES is not set, it will use the default /cfg directory.

You do not need to set the NMX_RES as long as you are using the /cfg directory for the version you are using. It is set automatically when you install the software.

The /cfg directory contains the following configuration files that are required for using Advanced Planning Agent:

File Name	Description
default.cfg	Pointers to Production Scheduling - Process, Strategic Network Optimization, online help, and so on. The default.cfg file contains the following configurable parameters: • <SCI_HOME_DIR>. The full path where Advanced Planning Agent is installed. • <SX_VER>. The Production Scheduling-Process version, if installed on the client machine. • <SX_HOME>. The full path where Production Scheduling-Process is installed. • <SX_LIB>. The name of the Production Scheduling-Process library directory. • <SX_EXEC>. The path, relative to SX_HOME, of the executable used to start Production Scheduling-Process. • <LX_VER>. The Strategic Network Optimization version, if installed on the client machine. • <LX_HOME>. The full path where Strategic Network Optimization is installed. • <LX_EXEC>. The path, relative to LX_HOME, of the executable used to start Strategic Network Optimization. • <LX_SMART_STRIP>. The path, relative to <LX_HOME>, where the Strategic Network Optimization lxsmartstrip.exe file is located. • <LX_DISP>. The path, relative to <LX_HOME>, where the Strategic Network Optimization Data Editor executable (DISP) is located. • <SCI_DISP_HOME>. The full path where the Advanced Planning Agent Data Editor is located. • <SCI_DISP_EXEC>. The name of the Advanced Planning Agent Data Editor executable, located in the <SCI_DISP_HOME> directory. • <SCI_LOGDIR>. The full path where you want Advanced Planning Agent to generate the progress log files and user log files. By default, this is set to /scilog, under the Advanced Planning Agent installation directory. • <SCI_LOG_SIZE>. The default maximum size for the Advanced Planning Agent log files. • <SCI_HELP_DIR>. The location, relative to <SCI_HOME_DIR>, of the Advanced Planning Agent help directory. • <SCI_HELP_FILE>. The Advanced Planning Agent help file name. • <SCI_ADMIN_HELP_DIR>. The location, relative to <SCI_HOME_DIR>, of the Advanced Planning Agent Administrator help directory. • <SCI_ADMIN_HELP_FILE>. The Advanced Planning Agent Administrator help file name. • <ERROR_BROWSER_HOME>. The location of the Advanced Planning Agent error browser application. • <ERROR_BROWSER_EXEC>. The Advanced Planning Agent error browser file name.

File Name	Description
default.cfg	• <SCI_RUNTIME_BINDINGS_EDITABLE>. Determines if you can edit a storage connection when you are running the Flow Manager with a runtime license. The valid values are “yes” and “no”. • <SCI_RUNTIME_DATAFLOWS_BROWSABLE>. Determines if you can browse data flows when you are running the Flow Manager with a runtime license. The valid values are “yes” and “no”. • <NOTE_PRINT_EXEC>. The print command name.
errmsg.cfg	Pointers to error logs, source text of error messages, and so on.
repository.cfg	Symbolic names of repositories and paths to their locations. Each repository name appears in angle brackets—for example, < reposABC > .You can set NMX_INIT to any of these names. The default value is default.

Note

All values in the default.cfg file are set during the installation process. Under normal circumstances, only the following entries should be modified:

- SX_*—Modify this entry if a new version of Production Scheduling-Discrete is installed after the Advanced Planning Agent installation.
- LX_*—Modify this entry if a new version of Strategic Network Optimization is installed after the Advanced Planning Agent installation.
- SCI_LOGDIR—Modify this entry to change the log file directory.
- SCI_LOG_SIZE—Modify this entry to change the log file size.

Windows Used to Maintain Configuration Files

Window Name	Navigation	Usage
Environment Info	Select Help, Environment in the Advanced Planning Agent Administrator or Manager window.	View system repository names.
UNIX command prompt	Enter the following command: echo $NMX_RES	View configuration file locations.
Change Info	Click Setup in the APAg Administrator or APAg Manager login window. Enter the configuration file directory name in the NMX_RES field and click OK.	Set configuration file locations.

Window Name	Navigation	Usage
UNIX command prompt	Enter the following command: `export NMX_RES=path/SCIdir/vers_x.x.x/cfg` where *x.x.x* represents the Advanced Planning Agent version number.	Set configuration file locations.

Updating Strategic Network Optimization and Production Scheduling - Process Versions in Configuration Files

To update Strategic Network Optimization and Production Scheduling - Process versions:

1. At a command prompt, change to the resource directory by entering the following command:

   ```
   cd $NMX_RES
   ```

   If the NMX_RES variable is not set, change to the directory path to which you usually export the NMX_RES variable.

2. Using a text editor, open the default .cfg file.
3. Edit the line containing the version number by replacing the old version number with the new one.

   For Production Scheduling - Process, this line is <SX_VERS> *x.x.x*. For Strategic Network Optimization, this line is <LX_VERS> *x.x.x*, where *x.x.x* is the version number of the software.

4. Ensure that the lines containing the location of Strategic Network Optimization and Production Scheduling - Process are correct.

   For Production Scheduling - Process, this line is <SX_HOME>. For Strategic Network Optimization, this line is <LX_HOME>.

5. Optional. For Strategic Network Optimization only. Edit the lines containing the Strategic Network Optimization executable and the smart strip by replacing the old version number with the new one. The executable line is:

   ```
   <LX_EXEC> bin/LX . x.x.x
   ```

   The smart strip line is:

   ```
   <LX_SMART_STRIP>bin/lxsmartstrip x.x.x
   ```

6. Save the default.cfg file.

   If the NMX_RES is not set up correctly or your repository is not listed in the repository.cfg file that you are using, the following error messages may appear:

   - Unable to initialize the resource area
   - Invalid configuration file
   - Login failed, check your name and password

Creating and Managing Storage Definitions

This section provides an overview of storage definitions, lists the windows used to create and manage storage definitions, and describes how to create storage definitions.

Understanding Storage Definitions

To connect Advanced Planning Agent to other systems and databases, you must create storage definitions. Storage definitions define information about databases and specify how Advanced Planning Agent data flows connect to databases. Storage definitions in Advanced Planning Agent have symbolic names; they are not the names of the databases. As a result, you can:

- Create multiple connections to the same database and each can have its own name.
- Give a connection a name that is more meaningful than the name of the database.

After you create storage definitions in Advanced Planning Agent, you can create storage connections that are linked to the storage definitions. Multiple projects can link to databases using the same storage definition.

After you create a storage definition, you must create at least one data object definition.

Windows Used to Create and Manage Storage Definitions

Window Name	Navigation	Usage
Storage Definition Manager	Select Manager, Storage Definitions in the Advanced Planning Agent Administrator.	View and manage storage definitions.
Create Storage Definition	In the Storage Definition Manager, click New.	Create storage definitions.
Edit Storage Definition	In the Storage Definition Manager, select a storage definition and click Edit.	Modify storage definitions.
Rename Storage Definition	In the Storage Definition Manager, select a storage definition and click Rename.	Rename storage definitions.
Storage Definition Manager	In the Storage Definition Manager, select a storage definition and click Delete. **Note** Deleting a storage definition in Advanced Planning Agent does not affect the database.	Delete storage definitions.

Creating Storage Definitions

Access the Storage Definition Manager.

To create a storage definition:

- Click New.
- In the first New Storage Definition window, enter the name for the new connection.

  The name must begin with an alphabetic character, and can include letters, numbers, and underscore (_) characters.

- Click OK.
- In the second New Storage Definition window, select one of the following options in the Type area:
  - Click relational database management system for a relational database
  - Click File for a flat file database.
- In the Sub-Type section, click the applicable relational database management system, or click token or fixed for flat files.
- In the Database field:
  - For a SQL relational database, enter the name of the database.
  - For an Oracle database, enter the service name for the database.

    Service names are defined in the */etc/tnsnames.ora file.*

  - For a flat file database, enter the directory path (it must already exist) or click Select to select the directory path.
- In the Connect field:
  - For SQL, enter the actual ODBC connection name.

    Make sure that there are no spaces in the name of the ODBC connection.

  - For other types, leave this field blank.
- In the Record field, enter the character that indicates the end of a record in the flat file database.

  The default is <newline> (the return character).

Note

For a flat file database, you must specify record, hard, and soft delimiters. You can also specify a comment delimiter.

- In the Hard field, enter the character that indicates the end of a field.

  The default character is <tab> (the tab character). For example, if the hard delimiter is the ^ character, the following record has three fields (a, b, and c):

  ```
  a^b^c^
  ```

  If nothing is between two hard delimiters, a null field is considered to be between them. For example, if the hard delimiter is the ^ character, the following record has five fields (a, null, null, b, and c):

```
a^^^b^c^
```

- In the Soft field, enter the character that indicates the end of a field.

  If a soft delimiter occurs between two soft delimiters, it is not null. In the line above, if the soft delimiter is ^, Advanced Planning Agent considers the line to have three fields: *a* , *b* , and *c* .

- In the Comment field, enter a character that identifies that the record is a comment.

  The default is *%%*, which is the same comment delimiter used in Production Scheduling - Process output files and Strategic Network Optimization output files.

- In the Description field, enter a brief description of the storage definition.
- Click OK.

Creating and Managing Data Object Definitions

This section provides an overview of data object definitions, lists windows used for creating and managing data object definitions, and discusses how to:

- Create data object definitions.
- Create data object definitions based on a database.
- Save data object definitions.
- Delete data object definitions.
- Print data object definitions.
- Edit data object field definitions.
- Edit data object field definition values.

Understanding Data Object Definitions

For each storage definition, you must create one or more data object definitions. Data object definitions specify the names of data tables in a database, the fields in the data tables, and the characteristics of each data table field.

If you want to create a data object that is the same as an existing relational database management system data table, you do not need to set up the definitions for the data object. Instead, you can use the definitions already in the database by using the Autoport option to create the data object definitions. You cannot use this Autoport option with flat file database tables.

A data object definition represents how the data is stored in each field of a data table. The characteristics of a field are displayed in one row in Advanced Planning Agent. The rows, automatically numbered by the system, are called field definitions.

In the data object definition, you can define whether a field is part of the primary key. Any number of fields can be chosen to be primary key fields. You can also define system keys if too many fields are primary keys or if no combination of fields can serve as a primary key. You cannot change any of the characteristics of a system key.

A data object name must be in a format that your database can use and can be up to 30 characters long.

If you are using an Oracle database, do not give the same name to more than one data object, even if you are using different cases. These databases are not case sensitive. For example, if you have a table named MYTABLE and you create a data object called MyTable, these databases consider the tables to have the same name, even though they use a different combination of uppercase and lowercase letters. When you save MyTable, Oracle overwrites MYTABLE, and you lose the data in MYTABLE. DB2 converts all the names of tables and fields to uppercase.

If you are using an Oracle, or DB2 database, and enter the field names, do not use Oracle, DB2, or DB2 keywords such as Max and From. If you use keywords, you receive an error message and are unable to save the data object.

You can verify data object definitions. When you verify data object definitions, a confirmation window appears if the definition is defined correctly. If a field definition contains an error, an error window appears. If a field definition contains an error, its text appears in red. If you select a field definition that contains an error, it is highlighted in red. You can view the errors in a selected field definition that contains an error.

Notes

A DB2 system key name can be a maximum of 18 characters long. The system key is generated by appending "_SK" to the table name. Therefore, the names of all DB2 tables requiring a syskey can be a maximum of 15 characters long.

When working with data object definitions in the APAg Administrator, you must respect the restrictions or limitations of any third-party system defined by the storage definition. For example, if you use an Oracle storage definition and Oracle has a restriction on the length of a field of a certain type, then your data object must respect that restriction. Advanced Planning Agent does not check to see if you comply with such restrictions, but it does return the errors produced by the third-party system to help you locate the problem.

In Windows, file locking may occur if you edit a data object definition while running a data flow.

For security reasons, do not use the storage definition sci_repository to define or store your data objects. This system repository should be used only for system-generated data objects. If you use the system repository for other data, you risk changing or deleting a system file, which could have serious consequences.

See Also

Mapping Data Types

Windows Used to Create and Manage Data Object Definitions

Window Name	Navigation	Usage
Edit Data Object	Select Manager, Data Definition in the APAg Administrator window. Select a data object definition and click Edit.	Edit data objects.

Window Name	Navigation	Usage
Rename Data Object	Select Manager, Data Definition in the APAg Administrator window. Select a data object definition and click Rename.	Rename data objects.
Print Table Definition	Access the Edit Data Object window. Select File, Print.	Print table definitions.
Edit Data Object	Access the Data Definition Manager window. Click Edit. Select Data Object, Verify.	Verify data objects.
Error Report	Access the Data Definition Manager window. Click Edit. Select View, Errors.	View data object definition error messages.

Creating Data Object Definitions

Access the Data Definition Manager.

You can create a data object definition whether the data table exists or not.

To create data object definitions:

1. In the Data Definition Manager, select a storage definition from the Storage Definition field.

   The data objects that exist for the storage definition appear in the Data Objects field.

2. Click New.
3. In the first New Data Object window, enter the name for the data object definition and click OK.
4. Complete the following fields for each data object in the table:
   - Key
   - Field name
   - Type
   - Width
   - Format
   - Description
5. Select File, Save.

See Also

Editing Data Object Definitions

Creating Data Object Definitions Based on a Database

Access the Data Definition Manager window.

To create data object definitions based on a database:

1. In the Data Definition Manager window, click Autoport.

   In the Autoport Native Objects window, all data tables in the database for which your relational database management system userid has permissions are displayed preceded by the userid.

2. Select a data table in the list and click OK.
3. In the Edit Data Object window, edit the data table and save the changes.

   You can save the data object definition with the same name or with a different name.

Note

If you select a data table in the Autoport Native Objects window that contains data types that are not supported by Advanced Planning Agent, the message Invalid_Type appears in the Type field.

Saving Data Object Definitions

Access the Edit Data Object window.

To save a data object definition:

1. Select File, Save As.
2. In the Save Data Object As window, select the storage definition that you want to use.
3. Do one of the following:
   - In the Name field, enter the new name for the data object definition.
   - In the Data Objects field, select a name and it automatically appears in the Name field.
4. Click OK.
5. (Optional) To create the physical data object in the database, click Yes.

   Note

   If you select Yes, the data in your data table is deleted. To avoid deleting any data, select No.

Deleting Data Object Definitions

Access the Data Definition Manager window.

To delete a data object definition:

1. Select a data object definition and click Delete.
2. In the confirmation window, select Yes.
3. (Optional) If the physical data table exists, do one of the following in the confirmation window that appears:

- If you want to delete the data table from the database, select Yes.
- If you want the data table to remain in the database, select No.

Printing Data Object Definitions

Access the Print Table Definition window.

Complete the following fields, and click Apply and Print:

Title	Title that you want to appear on each page.
Footer	Footer that you want to appear on each page.
Printer	Type of printer that you use. Values are: Postscript ASCII
Paper Size	Size of paper on which the report is printed.
Orientation	Select Landscape to print your report horizontally or select Portrait to print your report vertically.
Fit on One Page	Select this option if you want to print all of the information on one page. If your data can fill more than one page, the font size will be reduced so that all of your data fits onto one page.
Fit Width to Pages	Enter the number of pages for the report width if you do not want to condense the report to one page.
Fit Height to Pages	Enter the number of pages for the report height if you do not want to condense the report to one page.
Use Font Size	Enter the font size for your printed report. If you select the Fit on One Page option, the font size is ignored if it makes the report longer than one page.
Print Command	Print command. The default command is lp.

Edit Data Object Field Definitions

Access the Edit Data Object window.

To add, copy, paste, or delete a data object field definition:

1. Select a field definition.
2. Do one of the following:
   - To insert a field definition above the selected field definition, select Edit, Insert.
   - To cut the field definition so you can paste it in a new location, select Edit, Cut.
   - To copy the field definition so you can paste it in another location, select Edit, Copy.
   - To paste a field definition above the selected field definition, select Edit, Paste.
   - To delete the field definition, select Edit, Clear.

Editing Data Object Field Definition Values

Access the Edit Data Object window.

Complete the following fields:

Key Field that is part of a unique descriptor for each record in a data table. To change the value in this field, do one of the following:

- Select Key, Toggle Primary Key.
- Click the right mouse button and select Toggle Primary Key from the menu.

Values are:

Value	Description
Blank	The field is not a primary key or a system key.
Primary	The field is a primary key. A primary key indexes information in a database.
System	The field is a system key. Use a system key if too many fields are primary keys or if no combination of fields can serve as a primary key. You cannot change any characteristics of a system key.

Note. A DB2 system key name can be a maximum of 18 characters long. The system key is generated by appending "_SK" to the table name. Therefore, the names of all DB2 tables requiring a syskey can be a maximum of 15 characters long.

Field Name Name of the field.

Type Data type of the field. To select a data type, press and hold the left mouse button in the Type cell of a field definition and select a field type from the menu.

Values are:

Value	Description
Character	Relational databases differentiate between the character types CHAR and VARCHAR. Use the VARCHAR character type with Advanced Planning Agent because Advanced Planning Agent stores characters in this format. Although Advanced Planning Agent can convert the characters if you use CHAR, the conversion may be problematic.
Integer	A number with no decimal points. The greatest possible value for a field of this type is 214,783,647.
Smallint	An integer in the range -32768 to 32767. The smallint type uses less memory than the type integer.
Float	A number with decimal points. The exact number depends on your system.
Double	A number with more decimal points than float. The exact number depends on your system. Advanced Planning Agent internally uses the %g format to print this data type.
Date	A date in the format specified in the Format cell.
Syskey	System key data.

Text	The field contains binary or ASCII data such as an image or a pixel map.

Width Maximum number of characters that the field contains. If you also specify a format, the width must be the same as specified in the Format field. For example, if the format is %7.2f, the width must be 7. If you are using a fixed-format flat file storage definition, specify a width for every field definition. If you are using any other kind of storage definition, specify a width for every field definition with the character type. If you are using a date format, specify a width that is equal to or greater than the width of the date. For example, if you use the data format %y-%m-%d, specify a width no smaller than eight, which accounts for two characters for each number in the date and the two dash characters (-). For a character field, you must specify a width greater than zero. If the value in a character field is longer than its defined width in an relational database management system, the value is truncated on the right to fit the destination. For flat files, the value is not truncated.

Format Format of how data in the field is stored and displayed. You can specify:

- Number formats.
- Date formats.
- Character formats.

Number formats. A number format can convert, round, or append data so that the field is in the format that you require. You can use any number format available in ANSI C. This section explains only how to use the *f* formatDate formats. The syntax of the *f* format is as follows:

```
% xx.yf
```

Where:

- % and f are required.
- *xx* is the maximum number of digits in the numeric value, including the decimal point. If the specified width is greater than the number of digits in the data, the digits are right justified. This appearance is a display result only, which is only noticeable in a flat file.
- *y* defines the precision, which is the number of digits after the decimal point. Zeros are appended to the digits after the decimal point to equal the specified precision. If more digits exist than the specified precision, the number is reduced to the specified precision, and the last digit is rounded.

The following table shows the effects of various number formats:

Number	Format	How Number Appears
12.34	*%f (the default)*	*12.34 (Creates six digits after the decimal point)*
12.34	*%5f*	*12.34*
12.34	*%5.1f*	*_12.3*
12.345	*%.2f*	*12.35*
123456.78	*%10.2f*	*_123456.78*
1234.567	*%10.5f*	*1234.567*
1234.5678999	*%10.5f*	*1234.568*
In this table, the underscore character (_) represents a space.		

Note. For information about other format flags, refer to the sprintf section of a UNIX C manual or contact your system administrator.

Date formats. Values are:

Value	Description
%A	Weekday name. For example, Sunday, Monday, and Tuesday.

%a	Abbreviated weekday name. For example, Sun, Mon, and Tues.
%B	Month name. For example, January, February, and March.
%b	Abbreviated month name. For example, Jan, Feb, and Mar.
%d	Day of the month. For example, June 13 is stored as 13.
%H	Hour in a 24-hour clock. For example, 11:30 pm is stored as 23:30.
%I	Hour in a 12-hour clock. For example, 11:30 pm is stored as 11:30.
%j	Day of the year. For example, February 2 is stored as 33.
%m	Month as a number. For example, November is stored as 11 and March of the following year is stored as 15.
%M	Minute. For example, 20 minutes past the hour is stored as 20, or 15 minutes past the next hour is stored as 75.
%S	Second. For example, 20 seconds past the minute is stored as 20 or 15 seconds past the next minute is stored as 75.
%p	A.M or P.M.
%U	Number of the week in the year for a week starting on Sunday. For example, 17 represents the date of the seventeenth week in the year.
%W	Number of the week in the year for a week starting on Monday. For example, 17 represents the date of the seventeenth week in the year.
%Y	Year. For example, the year 2005 is stored as 2005. The year 0000, is treated as null.
%y	Year without the century. For example, the year 2005 is stored as 05.

You can specify characters such as spaces, hyphens, or slashes used in the stored dates as shown in the following examples:

Date Stored	Date Format
13 June 2005	*%d %B %Y*
2005-06-13	*%Y-%m-%d*
June 13, 2005	*%B %d, %Y*

Note: A value of fourteen consecutive zeros with a date format of *%Y%m%d%H%M%S* is treated as null.

Fixed: If you are using a fixed-format storage definition, your data object definitions use a fixed format. The syntax consists of three parts separated by a semicolon, as follows:

```
%#x;y;z
```

Where:

- *%#x* is a format indicator, which specifies the width and the kind of data.
- *y* is a fill character. If it is not specified, a space is used as the fill character. The fill character occupies any extra space in the fixed columns of a data field. For example, if you have a column for names that has a fixed width of 10, and a name has only 8 characters, the remaining two spaces in the column are filled with the fill character. The default fill character is a space.

- *z* is a justification character: *l* for left, *r* for right. The justification character indicates whether the information in the field is right justified or left justified before the fill characters are added. The character *r* specifies right justification, and the character *l* specifies left justification. For example, if you specify a fill character as a space and left justification, and you have eight characters in a column with a specified width of 10, the information is left justified and the fill characters occupy the ninth and tenth places:
  `12345678`
  If the same field was right justified, the fill characters would occupy the first two places:
  `12345678`

Each part (format indicator, fill character, and justification character) is optional. If it is specified, separate it from the next part with a semicolon. For example, the following format means that the data is a ten-character string (*10s*), the fill character is a slash (*/*), and the data is left justified (*l*): `%10s;/;l`

For character fields, you do not have to specify a width in the format indicator. You can specify the width of the character string in the Width field of the data object definition. For example, the following format means that the data is a string, the fill character is a slash (/), and the data is left justified (l): `%s;/;l or %c;/;1`

Values are:

Value	Description
%c	A character.
%d	A signed decimal (a number with a plus (+) or minus sign (-)).
%f	A signed decimal floating point.
%g *%G*	A signed decimal floating point. The format depends on the precision of the resulting value. For example, if the result has enough precision, it returns the number. If the result does not have enough precision, it uses scientific notation.
%e *%E*	A decimal floating point with the format displays one digit to the left of the decimal, a decimal, one (optional) digit to the right of the decimal, followed by e, a plus or minus sign, and at least two more decimal digits. Therefore `5.1e+3` represents the number 5100.
%s	A string.
%o	An octal.
%u	A decimal without a plus or minus sign.
%x	An unsigned hexadecimal.

Description Description of the field.

Connecting to Databases

This section provides an overview of databases and discusses how to:

- Establish relational properties using foreign keys.
- Map data types.
- Convert float data types to double data types.
- Connect to relational databases.

- Interface with DB2.
- Interface with Oracle.
- Interface with Microsoft SQL Server.
- Convert float data types to double data types.
- Map data types.

Understanding Databases

Although you do not require a detailed knowledge of databases to use Advanced Planning Agent, a general knowledge is helpful in understanding more fully how it works. If you intend to administer or build projects, you need a greater level of database knowledge and skills.

A database is a place where data is stored in data tables. You can use an ASCII flat file database (token-delimited or fixed-format) and the following kinds of relational databases: Oracle, DB2, and Microsoft SQL.

A data table consists of a heading and a body. The heading is a fixed set of fields (also called columns) and their data types. The body is a changing set of records (also called rows) in which all values correspond to the data types in the heading.

A flat file database is a directory that contains data tables with the data in readable text format, aligned in columns and typically separated by tabs. A flat file is a fast and flexible method for publishing data and is often the only form that some systems can use for importing or exporting data.

A relational database is managed by a relational database management system (relational database management system). The relational database management system presents data in tables and supports certain operations on data. A relational database provides additional features such as distributed access, a powerful data manipulation language called SQL, transaction control, and consistency control. A relational database provides stable, structured, and standardized storage and access of information. The data is available to multiple users, computers, and systems. On advanced systems, backup, restore, and administrative operations can be done in parallel with running systems. Systems from different vendors often support the same databases, making a physical integration of systems easy.

The relational database management system conforms to the principles and consistency standards of the relational model. Advanced Planning Agent can use a flat file database, a relational database, or a combination of databases.

Data tables contain:

- Data (the rows).
- Information defining the data format (the header).

  This information, called the data definition, includes the field names, the kind of data in each field, and the size of each field.

For example, a token-delimited flat-file database could contain the following fields and data:

Demand	Period	Product	Storage Location
3.000000	"Sep-05"	"APPLE12"	"AT-Apple-12"
1.400000	"Sep-05"	"APPLE12"	"BL-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"LA-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"MN-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"PL-Apple-12"
17.60000	"Sep-05"	"APPLE24"	"AT-Apple-24"
8.800000	"Sep-05"	"APPLE24"	"BL-Apple-24"
17.600000	"Sep-05"	"APPLE24"	"La-Apple-24"

This data table contains four fields: Demand, Period, Product, and Storage Location. The metadata is type double, date, char 30, char 30. The token delimiter is tab, and the record delimiter is carriage return. AT is Atlanta, BL is Baltimore, LA is Los Angeles, MN is Minneapolis, and PL is Portland.

Primary Keys

Some data tables have a primary key to uniquely define each record in the data table. The primary key is made up of one or more fields in the data table. Primary keys are assumed to be unique, which is actually enforced by relational database management system and must contain a value (they cannot be null).

For example, in the following data table, the primary key is the combination of the Period, Product, and Storage Location columns:

Demand	Period	Product	Storage Location
3.000000	"Sep-05"	"APPLE12"	"AT-Apple-12"
1.400000	"Sep-05"	"APPLE12"	"LA-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"MN-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"PL-Apple-12"
17.60000	"Sep-05"	"APPLE24"	"AT-Apple-24"
8.800000	"Sep-05"	"APPLE24"	"BL-Apple-24"
17.600000	"Sep-05"	"APPLE24"	"La-Apple-24"

When you select a primary key, ensure that each record of data is uniquely defined.

Field Definitions and Data Types

You must define the data type of each field so that Advanced Planning Agent can evaluate the data and perform calculations correctly.

The following table shows the available data types:

Data Type Name	Description
Character	Numbers and letters.

Data Type Name	Description
Date	A date in a specified format, such as YY/MM/DD or YYYY-MM-DD.
Double	A number with more decimal points of precision than float. The exact number depends on your system.
Float	Any number with any number of decimal places.
Integer	Any number with no decimal points.
Smallint	An integer in the range -32768 to 32767.
Syskey	An 8-byte data type used to uniquely identify records.

For example, in the following table, the Demand is type float, Period is type date, and Product and Storage Location are both type character:

Demand(FLOAT)	Period(DATE)	Product(CHAR)	Storage Location(CHAR)
3.000000	"Sep-05"	"APPLE12"	"AT-Apple-12"
1.400000	"Sep-05"	"APPLE12"	"BL-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"LA-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"MN-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"PL-Apple-12"
17.60000	"Sep-05"	"APPLE24"	"AT-Apple-24"
8.800000	"Sep-05"	"APPLE24"	"BL-Apple-24"
17.600000	"Sep-05"	"APPLE24"	"La-Apple-24"

Establishing Relational Properties Using Foreign Keys

A foreign key is a field or group of fields in a data table that refers to a primary key of some other data table. The field, or key, is shared between the tables. The key relates two tables together. Only a relational database can enforce foreign key constraints.

Foreign keys must either be wholly null or match some value of the corresponding primary key.

For example, the following employee table is an example of a parent table in a relational database:

Employee Data Table	Employee Skill Data Table
Employee ID	Employee ID
First Name	Programmer (y/n)
Last Name	Project Management (y/n)

The primary key of the Employee Skills Table is a foreign key and is used as a primary key in another table called the Employee Skills Table.

Mapping Data Types

If you use the Autoport option to create data objects using the definitions of an relational database management system data object, the relational database management system data types are mapped to Advanced Planning Agent data types. In other situations, Advanced Planning Agent data types must be mapped to relational database management system data types.

You can read and write data stored in packed decimal column types. Since these numbers are only formatted doubles or floats in Advanced Planning Agent, differences at reading time between stored numbers and numbers in Advanced Planning Agent could exist.

The width field is only used for Advanced Planning Agent character types, where it defines the maximum string length. The format field is used only for packed decimal numbers. Advanced Planning Agent retrieves these numbers as float or double. With reading, Advanced Planning Agent does not respect strict defined format.

If the value in a character field is longer than its defined width in an relational database management system, the value is truncated on the right to fit the destination. For flat files, the value is not truncated.

Converting Float Data Types to Double Data Types

In a data table, fields with float data types have precision limitations. For precision accuracy, you can define numbers that store values as double data types. For example, for greater precision accuracy in the result of an operation such as *PRODUCT($1.Demand,3.867)* , you can define the *Demand* field in the operation as double data type.

To prevent precision errors, you can convert all the fields of data tables in a storage definition of a repository from float data types to double data types.

The following format changes occur when you convert data to the Double data type:

- When only zeros appear after a decimal point, the number is written like an integer. For example, 1567.0000000000 is changed to 1567.
- Large numbers are written in exponential form. For example, 1234567.125000 is changed to 1.23457e+06.

To convert float data types to double data types:

1. In the SCIdir/vers_*x.x.x*/bin directory, enter floatToDouble.

   The current values of NMX_RES and NMX_INIT are displayed.

   To convert double data types to float data types, enter doubleToFloat.

2. At the prompt, confirm or change the settings.

   The name of the nmxcol file that is to be modified appears. The original file is backed up to the nmxcol.old file. If the backup is unsuccessful, you may still proceed with the conversion. A list of all storage definitions pertaining to the selected repository appears.

3. Do one of the following:

   - Enter the applicable storage number to select one or all storages to be modified.

- Enter q to quit the process.

4. If you select a single storage, the conversion takes place, and the list of storage definitions appears again for the next selection.

   A confirmation message appears when all or q is selected, and the process is complete.

Connecting to Relational Databases

The following technical information describes how Advanced Planning Agent can connect to your relational database management system. For additional information, refer to the administration guides and manuals applicable for your relational database management system.

The version number of a supported relational database management system is subject to change. The version number is the minimum supported by Advanced Planning Agent, although other versions may also work.

Interfacing with DB2

The following table provides information about interfacing with DB2:

Requirement	Description
Required Components	• DB2 Server v2.1.1 or DDCS 2.3 (if connecting to a DRDA compliant database; DDCS includes the CAE component) • Client Application Enabler (CAE) • If running on IBM, the minimum O/S version is AIX 4.1 • If running on HP, the minimum O/S version is HP-UX 10.20
DB2 Shared Libraries	`libdb2.sl` (for HP) and `libdb2.a` (for IBM) shared libraries. During installation, the corresponding library is placed under the Advanced Planning Agent version directory. For example, if you install version 8.10 of Advanced Planning Agent on the HP, then `libdb2.sl` is located under `SCIdir/vers_8.10`. For this library to be visible (in case your site does not have its own version of the library), add the above path to the SHLIB_PATH (for HP) and LIBPATH (for IBM) environment variable.

The following table describes the data type mapping from Advanced Planning Agent to DB2:

Advanced Planning Agent Data Type	Advanced Planning Agent Format	DB2 Data Type
Syskey		CHARACTER(16)
Character(size)		VARCHAR(size)
Date		TIMESTAMP(10)
Double		DOUBLE(8)
Float		DOUBLE(8)
Integer		INTEGER(4)

Advanced Planning Agent Data Type	Advanced Planning Agent Format	DB2 Data Type
Smallint		INTEGER(4)
Double	%p.sf	DECIMAL(p,s)
Float	%p.sf	DECIMAL(p,s)
Text		CLOB(size)

The following table describes the data type mapping from DB2 to Advanced Planning Agent:

DB2 Data Type	Advanced Planning Agent Format	Advanced Planning Agent Data Type
CHAR(size)		character(size)
VARCHAR(size)		character(size)
TIMESTAMP		Date
DOUBLE		Double
DECIMAL		Float
INTEGER		Integer
SMALLINT		Integer
DECIMAL(p,s) s=0		Integer
DECIMAL(p,s) s≠0		Float
DECIMAL(p,s)	%p,sf	Double
DECIMAL(p,s)	%p,sf	Float
CLOB(size)		Text

DB2 converts all of the names of tables and fields to uppercase. The DB2 data type sizes are in bytes. The DB2 data types LONG VARCHAR, DATE, TIME, BLOB(size), DBCLOB, GRAPHIC(size), VARGRAPHIC(size), and LONG VARGRAPHIC are not supported.

Interfacing with Oracle

The following components are required to interface with Oracle:

- Oracle database
- SQLnet version 2 on the Oracle Server's side (this is usually installed along with the Oracle Server)
- TCP/IP

Once all the above components are in place, you must set up the tnsnames.ora file. This file is an Oracle file, used by the Oracle libraries compiled in Advanced Planning Agent to determine the

location of the database to which you require a connection. The default directory to place this file is /etc. If, however, a requirement exists to place this file under a different directory, then that directory must be the value for the environment variable called TNS_ADMIN. The value for this variable must be exported before running Advanced Planning Agent. For example, if you place your tnsnames.ora file under /home/Oracle/env, then each time that you run Advanced Planning Agent, you must do the following:

```
export TNS_ADMIN=/home/Oracle/env
```

If this variable is not set up properly, a connection from Advanced Planning Agent to Oracle is not possible. This procedure is standard for all third-party systems that offer Oracle connectivity. A sample entry in the tnsname.ora file could look like the following:

```
database_alias =
   (DESCRIPTION =
   (ADDRESS =
     (PROTOCOL = TCP)
      (HOST = < hostname >)
      (PORT = 1521)
   )

   (CONNECT_DATA =
   (SID = < db_name >)
   )
)
```

The *database_alias* is the alias for the database. You place this value in the Database field when creating an Advanced Planning Agent storage definition for the specific Oracle database or when trying to connect to Oracle with sqlplus.

The following table describes the data type mapping from Advanced Planning Agent to Oracle:

Advanced Planning Agent Data Type	Advanced Planning Agent Format	Oracle Data Type
Syskey		RAW(8)
character(size)		VARCHAR(size)
Date		DATE
Double		FLOAT(126)
Float		FLOAT(126)
Integer		NUMBER(38)
Smallint		NUMBER(38)
Text		LONG
Double	%p.sf	NUMBER(p,s)
Float		NUMBER(p,s)

The following table describes the data type mapping from Oracle to Advanced Planning Agent:

Oracle Data Type	Advanced Planning Agent Format	Advanced Planning Agent Data Type
CHAR(size)		character(size)
VARCHAR2(size)		character(size)
NUMBER(p,s)		Double
FLOAT(p)		Double
DATE		Date
LONG		Text
RAW(8)		Syskey
RAW(size), size≠8		Character
NUMBER(p,s)	%p,sf	Double
NUMBER(p,s)	%p,sf	Float

The Oracle data types LONG RAW, ROWID, and MLSLABEL are not supported. You can use ANSI/ISO, DB2, and SQL/DS data types in Oracle, but they are internally converted to Oracle data types.

When you use Oracle tools to populate Oracle tables after the year 2000, using the 2-digit year format, Oracle reads the year as a year in the 21st century. For example, if you insert a record such as 15-Aug-95, Oracle reads this date as 15-08-2095. You must use Oracle's RR date format if you want to use the 2-digit year format. For more information, see Oracle documentation.

Note

This interpretation of dates is an Oracle characteristic and not Advanced Planning Agent. If you use Advanced Planning Agent to write a date such as 15-Aug-95, the date is written in Oracle as 1995 because Advanced Planning Agent writes a 4-digit format to Oracle.

Interfacing with Microsoft SQL Server

The following components are required to interface with Microsoft SQL Server:

- ODBC SQL Driver – recommends Version 3.5. The driver is installed automatically with SQL client software.
- Microsoft® SQL Server – recommends Version 6.5 or later. The server communicates with the network software through the SQL Server Net library.
- Network Software.

Connecting to the SQL Server

You must create an ODBC data source after the SQL Server client is installed and configured.

To create a data source:

1. From the ODBC SQL Server Setup dialog box, add a data source from which you want to connect.
2. From the ODBC Driver list, select SQL Server Driver.
3. When the data source is created, you can verify the connection.

The following table describes the data type mapping from Advanced Planning Agent to MS SQL:

Advanced Planning Agent Data type	Advanced Planning Agent Format	MS SQL Data type
Syskey		BINARY(8)
Character(size)		VARCHAR(size)
Date		DATETIME
Double		REAL
Float		FLOAT
Integer		INT
Smallint		INT
Double		DECIMAL(p,s)
Float	%p,sf	DECIMAL(p,s)
Text	%p,sf	TEXT

The following table describes the data type mapping from MS SQL to Advanced Planning Agent:

MS SQL Data Type	Advanced Planning Agent Format	Advanced Planning Agent Data Type
CHAR(size)		character(size)
VARCHAR(size)		character(size)
NCHAR(size		character(size)
NVARCHAR(size)		character(size)
BINARY		Syskey
VARBINARY		Syskey
DATETIME		Date
SMALLDATETIME		Date
FLOAT		Float
REAL		Double
INT		Integer

MS SQL Data Type	Advanced Planning Agent Format	Advanced Planning Agent Data Type
TINYINT		Integer
NUMERIC(p,s)	%p.sf	Double
NUMERIC(p,s)	%p.sf	Float
DECIMAL(p,s)	%p.sf	Double
DECIMAL(p,s)	%p.sf	Float
TEXT		Text

The Microsoft SQL Server data types SMALLMONEY, MONEY, CURSOR, TIMESTAMP, IMAGE, BIT, UNIQUEIDENTIFIER are not supported.

Managing Advanced Planning Agent Users

This chapter provides an overview of read and write permissions, lists the windows used to manage Advanced Planning Agent users and discusses how to:

- Manage user access rights.
- Set browse permissions.

Understanding Read and Write Permissions

A user must have read and write permissions to the user's own home directory.

To view or change the data in a flat file database, you must have read and write permissions to the database directory and files.To use a flat file repository, you also must have read and write permissions to the repository and its contents.

To view or change the data in a relational database or repository (Oracle or DB2), you must have the correct permissions in the relational database. These permissions are set up by the database administrator.

Windows Used to Manage Advanced Planning Agent Users

Window Name	Navigation	Usage
New User	Select Manager, User in the APAg Administrator window. Click New in the Users Manager window.	Create user accounts.
Rename User	Select Manager, User in the APAg Administrator window. Click Rename in the Users Manager window.	Rename user accounts.

Window Name	Navigation	Usage
Users Manager	Select Manager, User in the APAg Administrator window. Select a user and click Delete.	Delete user accounts.
Edit User	Select Manager, User in the APAg Administrator window. Select a user in the Users Manager window and click Edit. **Note** After you change a user password, you must update the user's access rights to storage definitions.	Change user passwords.
Access Rights Manager	Select Manager, Access Rights in the Advanced Planning Administrator window.	Manage user access to databases.

Managing User Access Rights

Access the Access Rights Manager window.

The Access Rights Manager displays the users and the storage definition. The selected user has access to the storage definitions that are highlighted.

To manage user access rights:

1. In the Users field, select a user.
2. In the Storage Definition field, select the storage definitions to which the user requires access.

   To take away access to a storage definition, select a highlighted storage definition to remove the highlighting.

3. Click Update.
4. In the confirmation window, click Yes to assign the access rights.
5. (Optional) If you have selected one or more relational database management system storage definitions, a DataBase User Info window appears for each relational database management system, one at a time.

   Enter the user name and password of the user for each database. The data objects associated with the storage definition are created in the schema of the user's database.

6. Click Access Info to display the Access User ID's window.

   The login ID that the user has for each relational database management system storage definition is displayed.

Setting Browse Permissions

To set browse permissions, do one of the following:

- To allow runtime users to browse data flows in the runtime version, ensure that the following appears in the default cfg file:

  ```
  <SCI_RUNTIME_BINDINGS_BROWSABLE> yes
  ```

- To prevent users from browsing data flows in the runtime version, ensure that the following appears in the default cfg file:

  ```
  <SCI_RUNTIME_BINDINGS_BROWSABLE> no
  ```

Creating and Managing Advanced Planning Agent Projects

Using the Advanced Planning Agent Manager, you can create projects that contain information about database connections and data flows. This chapter discusses how to:

- Start the Advanced Planning Agent Manager.
- Create and open projects.
- Specify storage connections and data objects.

Starting the Advanced Planning Agent Manager

This section discusses how to:

- Start the Advanced Planning Agent Manager in Windows.
- Start the Advanced Planning Agent Manager from a command line in Windows.
- Start the Advanced Planning Agent Manager in UNIX.

Starting the Advanced Planning Agent Manager in Windows

To start the Advanced Planning Agent Manager in Windows:

1. Login with your userid.
2. Select Start, Programs, Supply Chain Planning, Advanced Planning Agent *x.x.x*, Advanced Planning Agent Manager, where *x.x.x* is the Advanced Planning Agent version that you want to run.
3. In the Login window, enter your userid in the Name field.
4. In the Password field, enter your password.

   Note

   The default userid of the demo repository that is shipped with Advanced Planning Agent is nmx_admin. The password is admin. Each repository has its own password for the nmx_admin user. You can specify the password for the nmx_admin user when you create a repository.

5. Click OK.

   Note

   If you cannot login, ensure that the correct system repository is selected.

Starting the Advanced Planning Agent Manager from a Command Line in Windows

To start the Advanced Planning Agent Manager from a command line in Windows:

1. Open a command prompt window.
2. Change to the APS/bin directory.
3. Enter the following command:

```
apag [-display hostname:0:0]
```

4. Enter 2.
5. Enter the number for the Advanced Planning Agent version that you want to run.

Starting the Advanced Planning Agent Manager in UNIX

To start the Advanced Planning Agent Manager in UNIX:

1. In a UNIX window, enter apag.

   A list of options appears.

2. Type 2 and press Enter.

   A list of installed versions appears.

3. Select the version that you want to use.

   Do not close the UNIX window because Advanced Planning Agent will stop immediately.

4. In the Login window, enter your userid in the Name field.

   The default userid is nmx_admin. Check with your system administrator if you need your own userid and password.

5. In the Password field, enter your password.

   Note

   The default userid of the demo repository that is shipped with Advanced Planning Agent is nmx_admin. The password is admin. Each repository has its own password for the nmx_admin user. You can specify the password for the nmx_admin user when you create a repository.

6. Click OK.

Note

If you cannot login, ensure that the correct system repository is selected.

Creating and Opening Projects

Projects in Advanced Planning Agent contain information about database connections and data flows. This section lists the windows used to create and open projects and discusses how to:

- Refresh project path names in the Data Flow editor.
- Match patterns using wildcard characters.

Windows Used to Create and Open Projects

Window Name	Navigation	Usage
Project	Select File, New in the APAg Manager window.	Create projects.
Open Project	Select File, Open in the APAg Manager window. **Note** The project name appears in the title bar of the Advanced Planning Agent Manager window. Advanced Planning Agent is shipped with a sample project called Granola.prj.	Open projects.
Save Project	Select File, Save As in the APAg Manager window.	Save a project with a new name or location.

Refreshing Project Path Names in the Data Flow Editor

If the project path name in the data flow editor does not refresh correctly, you must update Hummingbird Exceed files.

To update Hummingbird Exceed files:

1. Select Start, Programs, Exceed Hummingbird Update.
2. In the Hummingbird Update window, click Update Now.
3. In the Checking for Update Files window, click OK.

Note

You must use Exceed 6.2 or a later version.

Matching Patterns Using Wildcard Characters

The following table shows wildcard characters that you can use in system windows to match specified patterns:

Wildcard Character	Description
*	Match any sequence of characters.
?	Match any single character.
\	The next character does not have a symbolic meaning. For example, \? means that the ? is part of the text and not a wildcard character.
[ ]	Match any character enclosed in the square brackets.
-	Match a range of characters separated by this character; works only within square brackets.

Specifying Storage Connections and Data Objects

This section provides an overview of storage connections and data objects, lists the windows used to create and manage storage connections and data objects, and discusses how to:

- Create storage connections.
- Set parameters for storage connections.

Understanding Storage Connections and Data Objects

Storage connections are links to the storage definitions that you create using the Advanced Planning Agent Administrator. You can create as many storage connections as you need and you link multiple storage connections to the same storage definition. Before you can create a storage connection, the storage definition that it uses must exist.

The Default Data Objects option simplifies the creation of storage connections. When you select this option, Advanced Planning Agent uses the table data defined in the Advanced Planning Agent Administrator and creates a project with the same name that includes all of the data tables in the selected storage definition. When you define a storage connection, you do not need to define each data object in Advanced Planning Agent that already exists in your database.

You can rename and delete storage connections. Deleting a storage connection does not affect the storage definitions created by the Advanced Planning Agent Administrator.

A data object represents a data table. The Data Object Manager links a data object to a storage connection and a data table. A data object is defined by its storage connection and the data table to which it is linked.

You can use the Data Object Manager to perform the following tasks:

- Creating a data object.

- Renaming a data object.
- Deleting a data object.
- Changing a storage connection for a data object.

Windows Used to Create and Manage Storage Connections and Data Objects

Window Name	Navigation	Usage
New Storage Connection	Select Manager, Storage Connection in the APAg Manager window. Click New in the Storage Connection Manager window.	Create storage connections.
Rename Storage Connection	Select Manager, Storage Connection in the APAg Manager window. Select a storage connection in the Storage Connection Manager window and click Rename.	Rename storage connections.
Storage	Select Manager, Storage Connection in the APAg Manager window. Select a storage connection in the Storage Connection Manager window and click Delete.	Delete storage connections.
Rename Data Object	In the Data Object Manager window, select a data object and click Rename. In the Rename Data Object window, enter a new name for the data object and click OK. Click Yes to change the name of the data object within all data flows in the project.	Rename data objects.

Window Name	Navigation	Usage
Data Object Definition	In the Data Object Manager window, select a data object and select Edit. In the Edit Data Object window, from the Storage Connection field, select the storage connection to which the data object will be linked. The data tables for the storage connection appear in the Attached To field. In the Data Object Definition field, select the data table to which the data object will be linked.	Change the storage connection and data table to which a data object is linked.
Data Object Manager	Select Manager, Data Object in the Advanced Planning Agent Manager. In the Data Object Manager window, select a data object. Click Delete. In the confirmation window, click OK. **Note** Deleting a data object does not affect the storage connections or data tables.	Deleting Data Objects

Creating Storage Connections

Access the New Storage Connection window.

To create a storage connection, complete the following fields:

Storage Definitions Select the storage definition to which the storage connection will be linked.

Default Data Objects Click this option to create default data objects.

Setting Parameters for Storage Connections

To allow runtime users to create and edit storage connections, ensure that the following appears in the default .cfg file:

```
<SCI_RUNTIME_BINDINGS_EDITABLE> yes
```

If you do not want users to edit storage connections, set the parameter to no.

Creating and Managing Data Flows

This chapter provides an overview of data flows and discusses how to create and manage data flows.

Understanding Data Flows

To synchronize your supply chain systems, you can create data flows in the Advanced Planning Agent Manager. Data flows represent the flow of information across the collaborative enterprise network, and are used to integrate data from disparate systems into one system. For example, you can create a data flow that extracts data from one system and manipulates the data using filters to prepare the information for use in another system.

Data flows consist of nodes connected by arcs which imply a sequential flow of data execution.

Nodes

Each node in a data flow represents a data table or performs an action on the data. When you run a data flow, the starting data passes through the nodes that make up the data flow from left to right. The data is moved, manipulated, and optionally displayed as each node runs.

After you create a data flow, you can add nodes and arcs. When you create a node, you can provide a symbolic name. For example, if the name of a data table is Int_inv_Pt, you might use the symbolic name Initial_Portland_Inventory to illustrate the purpose of the data table. Each node has an icon that represents the type of node and a symbolic name.

The following table lists the types of nodes that you can create in data flows:

Name	Icon	Description
Data object		Represents data in a data table.
Selection filter		Selects information from input data set.
Aggregation filter		Groups related records in a data table into one record. Groups discrete events and their dates into time buckets.
Join filter		Unites data from two input data sets into one.
Data Editor		Performs calculations and produces reports with the Data Editor.
Production Scheduling - Process		Connects to the Production Scheduling - Process system.
Strategic Network Optimization		Connects to the Strategic Network Optimization system.
Shell		Runs a UNIX script, such as sed or awk, or a Tcl script.
Data flow		Embeds a data flow in another data flow.
Input		Represents the data that flows from a data object node in the superflow to the subflow.
Output		Represents the data that flows from a subflow to a data object node in the superflow.

You can add comments to nodes in data flows. You can use these comments to do the following:

- Describe the purposes of each node.
- Indicate when each node was last edited.
- Provide instructions or rules about how and when to use the data flow.

Arcs and Attach Points

You need to connect nodes with arcs for data to flow from one node to another, and to indicate in which order the nodes are run.

Arcs and attach points show the flow of data from one node to another, or the dependency of one node on another. You need to connect nodes with an arc for data to flow from one node to another, and to indicate in which order the nodes run. Arcs attached between right and left attach points indicate a data flow. Nodes attached at the top or left of a node run before the nodes attached to the right or bottom.

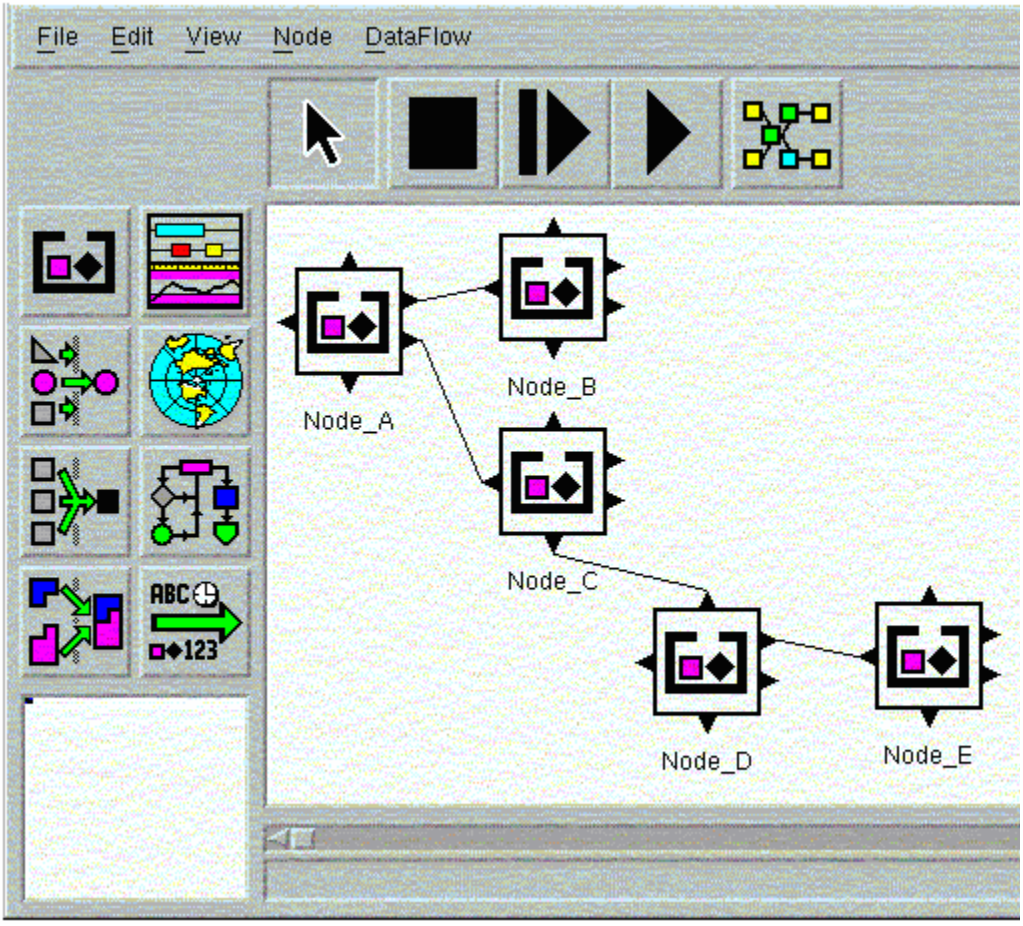
Attach points are the small black objects on the edges of a node. The number of attach points depends on the kind of node. The attach points can be on any side of a node. The position of the attach point means the following:

Position	Meaning
Left side	Input to the node.
Right side	Output from the node.
Top	The node attached to the top is run before the node.
Bottom	The node attached to the bottom is run after the node.

The data flow goes from left to right across the model, and top to bottom where necessary. When you run a data flow, the node on the left end of an arc runs before the node on the right. If nodes are attached vertically, the flow moves from top to bottom.

For example, in the following data flow:

- Node A runs first because it is on the left end of the arcs that connect it to Node B and Node C.
- Node B and Node C are not connected so they can run in any order.
- Node C and Node D are connected and Node D is dependent on Node C. Node C runs before Node D.
- Node D runs before Node E because it is on the left end of the arc that connects them.



Data flow

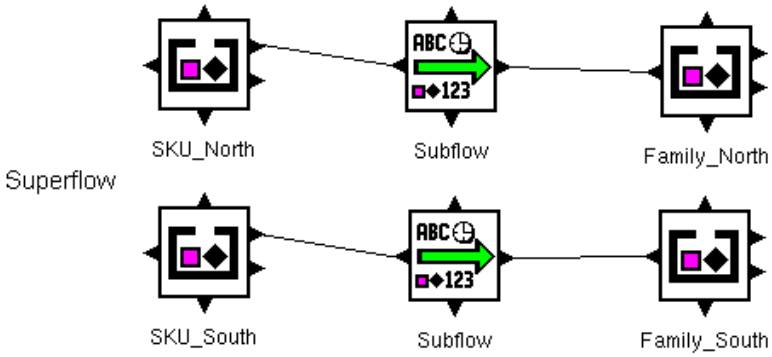
Subflows

You can create a data flow that is used in other data flows to act as a single node. This kind of data flow is called a subflow. A data flow that contains a subflow is called a superflow.

Subflows can save you time and effort because one subflow representing a common group of operations can be used in any other data flow. A subflow is represented by a data flow node in the superflow. Input nodes and output nodes show how data is passed between the superflow and the subflow.

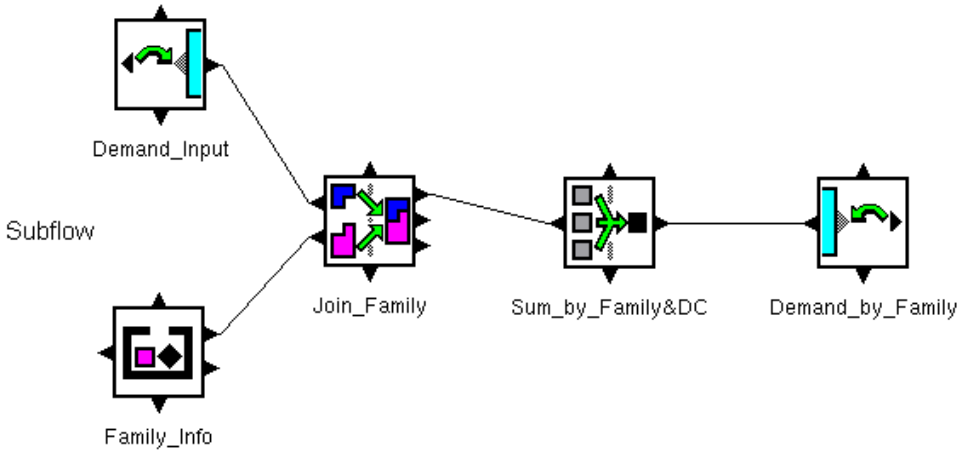
You must use input nodes in a subflow so that it can receive data from the superflow. The input nodes connect data object nodes in the superflow to the subflow, supplying the subflow with data.

The following graphic shows a superflow:



Superflow

The following graphic shows a subflow:



Subflow

For example, the first data flow is a superflow because it contains a subflow. In fact, it contains the same subflow twice. The second data flow is a subflow because it consists of common operations that are used in the superflow. The same subflow can be used in any other superflow.

You can use the following nodes when creating subflows:

Node Type	Icon	Description
Data flow		Embeds a data flow (the subflow) in another data flow (the superflow).

Node Type	Icon	Description
Input		Represents the data that flows from a data object node in the superflow to the subflow. You create input nodes in the subflow. You use one input node for each data object node in the superflow that sends data to the subflow. You must use input nodes in a subflow so that it can receive data from the superflow. When the subflow is embedded in the superflow, the input nodes are represented by the attach points on the left of the subflow node. The input nodes connect data object nodes in the superflow to the subflow, supplying the subflow with data. The number of attach points on the left side of a data flow node equals the number of data object nodes that inputs data into a subflow. The input nodes rely on data tables, and subflow operations must use the same fields as the data tables. For example, if a data table contains the fields Family, Data, and Quantity, the subflow must be set up to use the same fields.
Output		Represents the data that flows from a subflow to a data object node in the superflow. A nested flow can provide output data. In this situation, you need to use output nodes in the subflow. When the subflow is nested in a second flow, its output attach points match the output nodes used in the second subflow. Each output attach point must be connected to a node that uses the appropriately named fields produced by the second subflow. You use one output node for each data object node in the superflow that receives data from the subflow. You must ensure that the subflow operations use the same fields as the data tables. For example, if the output data from a subflow operation outputs information about Family, Date, and Quantity, the node using this data must use the fields of Family, Date, and Quantity. When the subflow is embedded in the superflow, the output nodes are represented by the attach points on the right of the subflow node.

A nested flow can be used to create a dependency. In this case, you do not need to use the input and output data nodes in the subflow. If you do not use input and output nodes in the subflow, it does not have input and output attach points (on the left and right of the node) when it is nested in the superflow. However, a subflow always has dependency attach points (on the top and bottom of the node).

If the subflow has input nodes or output nodes, the data flow node is created with the correct number of attach points on its left and right. Connect the attach points to their respective data object nodes.

When you run a data flow containing a subflow, the subflow also runs. You do not have to open a subflow for it to run.

Verifying Data Flows

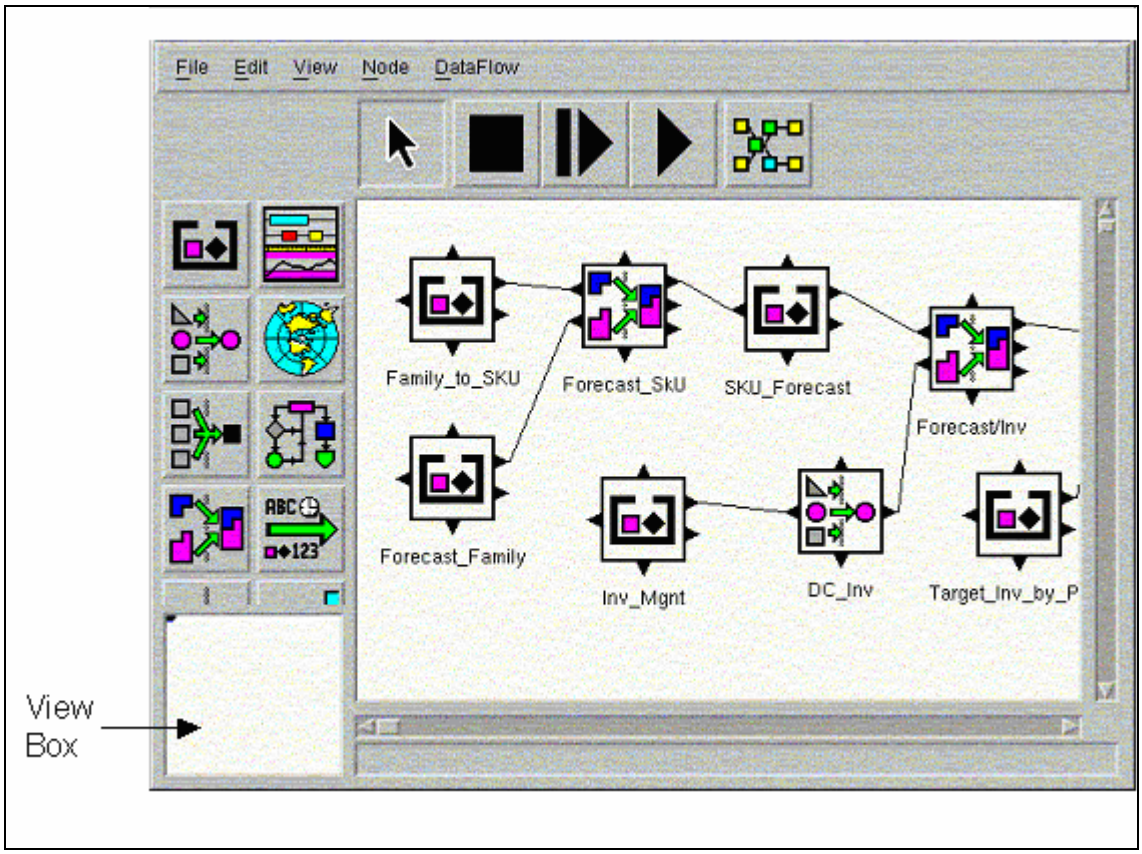
When verifying, the data flow is checked to ensure that the connections between nodes are appropriate; the syntax of expressions is correct; and the files specified in the Data Editor node, Strategic Network Optimization node, and Production Scheduling - Process node exist.

If an error is detected, a warning window appears, and every node where an error is detected has its name highlighted in red.

Even if no errors are found, the data flow may not be correct in all respects. Some logical errors, for example, can only be detected when a data flow runs. You must ensure that what a data flow does is sensible and suits your requirements.

Viewing Different Areas of the Data Flow Workspace

The view box displays the position of the data flow with respect to the entire available workspace. The small rectangle in the view box indicates the visible portion of the workspace and correlates the workspace with the size of the window. In the view box, click and drag the square to the area of the workspace that you want to view.



View box

Creating and Managing Data Flows

This section lists windows used to create and manage data flows and discusses how to:

- Create nodes in data flows.
- Move nodes in data flows.
- Align nodes in data flows.
- Connect nodes with arcs.
- Add subflows to data flows.

Windows Used to Create and Manage Data Flows

Window Name	Navigation	Usage
Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit.	View or edit data flows.
Data Flow - New	Select Manager, Data Flow in the APAg Manager project window. Click New.	Create data flows.
Rename Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Rename.	Rename data flows.
Delete Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Delete.	Delete data flows
Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Select DataFlow, Verify.	Verify data flows.
Note Browser	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Select View, Browse Notes.	View comments associated with nodes.
Note Editor	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Click a node and select Edit, Notes.	Add comments to notes.

Creating Nodes in Data Flows

Access the Data Flow window.

To create a node in a data flow:

1. Do one of the following:
   - Click a node in the node bar.
   - Select a node from the Node menu.
2. Do one of the following:
   - To create one node of the selected type, click in the data flow area.
   - To create multiple nodes of the selected type, hold the Ctrl key and click in the data flow editor.

     Each time that you click, a node of the same kind is created.

Moving Nodes in Data Flows

Access the Data Flow window.

To move nodes in a data flow, do one of the following:

- To move one node, click and drag the node to the new location.
- To move a group of nodes, draw a box around the nodes and and drag the group to the new location.

Aligning Nodes in Data Flows

Access the Data Flow window.

To align nodes in data flows, select the nodes that you want to align and do one of the following:

- To align the nodes horizontally, select Edit, Align, Row.
- To align the nodes vertically, select Edit, Align, Column.

Connecting Nodes with Arcs

Access the Data Flow window:

To connect nodes with an arc, click the attach point on one node, and drag an arc to an attach point on another node.

Note

You cannot draw an arc between incompatible nodes or attach points.

Adding Subflows to Data Flows

Access the Data Flow window.

To add a subflow to a data flow:

1. Create or open a data flow that you want to use as a subflow.
2. Create or open the data flow that you want to use as the superflow.
3. Create a data flow node in the position where you want to embed the subflow.
4. Select Edit, Properties.
5. Select the data flow that you want as a subflow and click OK.

Setting up Data Object Nodes

Each data flow in Advanced Planning Agent contains numerous data object nodes.

Data object nodes perform the following functions:

- Save selected data changes to a data table.
- Read selected data from a data table.
- Browse data in a data table.

This chapter provides overviews of data object node options, handling exception output from data object nodes, and viewing and editing data in data object nodes, lists the windows used to view and manage data object nodes, and discusses how to view selection criteria fields and functions.

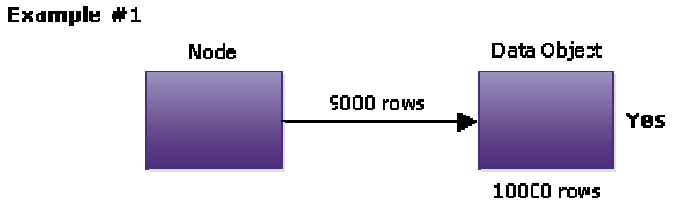
Understanding Data Object Node Options

To use a data object node, you must set up its options. Data object node options include the following:

- Copy By Name option. When you run a data flow, data from input data object nodes is copied to output data object nodes. Currently, the following two modes exist for the copy option:
  - The first mode copies by position and is the default. In this case, the input data records and the output data records must contain fields of the same type in the same position. Data in corresponding positions is copied from input to output data records. Fields with the same names must have the same type.
  - The second mode copies by the field names. Fields contained in the input data records and not in the output data records are ignored. Fields contained in the output data records but not in the input data records are given the value of null.
- Editable Option. You can browse and edit the data table that a data object node represents in the Data Editor. Click Editable if you want to edit and save data in the Data Editor. The editable option is used mainly in run mode to control the browse permissions for run-only users.
- Read Before Write option. The Read Before Write option can increase the speed of a flow for relational database management systems (relational database management system), depending on the size of the input data and the size of the data table that is being read.

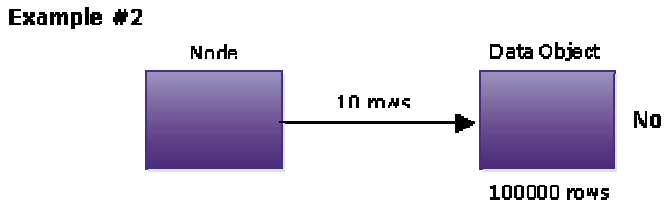
  The following diagrams show examples of the Read Before Write option. The first two of examples are situations for which the Read Before Write should and should not be used. The third example is a sample situation that is questionable and requires trial runs.

  In the first example, the number of rows to be written to the original table is large relative to the size of the original table. In this case, the write is faster if the original table is read into memory, so turn on the Read Before Write option. The following diagram illustrates this example:



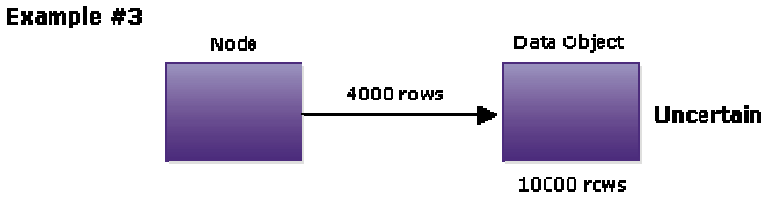
Example: Use theRead Before Write option

In the second example, the number of rows to be written to the original table is small relative to the size of the original table. In this case it is faster to write each row to the database and not read all of the data in the table into memory, so turn off the Read Before Write option. The following diagram illustrates this example:



Example: Do not use the Read Before Write option

In the third example, the number of rows to be written to the original table is approximately half the size of the original table. It is uncertain which mode is faster and you need to test both modes of the Read Before Write option. The following diagram illustrates this example:



Example: Need to test the Read Before Write option

With relational database management systems, when a record is to be written, the Advanced Planning Agent Manager module first looks for it in memory. If it is found in memory, then it is changed in memory and an update is issued to the data base. If it is not found in memory and all relevant data is already in memory, an insert is issued to the data base. If all relevant data is not already in memory, the Manager module issues a query to the database to decide if inserting a new record or updating an existing record is necessary. This extra query is very slow to process for some databases. If the Manager module reads in the data before it writes, then all the relevant data is in memory and the extra query is avoided.

This situation can be controlled by the Read Before Write option. Setting the option to "on" forces data to be read in before writing. This option currently has no effect on flat file data object nodes. It also has no effect on data object nodes with output because data is always read in for them. The Read Before Write option has no effect if the Delete/Write option is set on.

With flat file databases, when data is needed from a flat file, the entire data file is always read into memory, regardless of any applicable selection criteria. Once the data table is in memory, the selection criteria are applied and the subset data that satisfies the selection criteria is returned. Therefore, the Read Before Write option does not affect flat files.

- Overwrite option. The Overwrite option overwrites a table in the output node with a table from the input node as follows:
  - If a record from the input table matches a record in the output table, the input record replaces the record in the output table.
  - If a record in the input table does not exist in the output table, it is added to the output table.
  - Records in the output table that have no match to records from the input table remain unchanged.
  - If no primary key is defined in the data object definition for the output node, then the records of the input table are appended to the output table and no changes to the existing records of the output table are made.
- Delete/Write option. The Delete/Write option deletes the data in the output table and continues as in the Overwrite option.

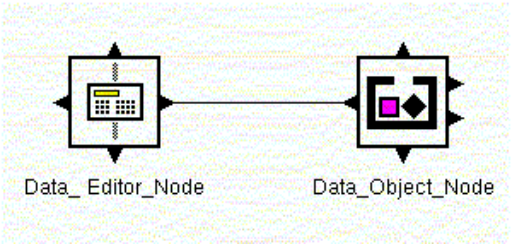
  In the following table, when the Delete/Write option is turned on for NodeTwo, the data in NodeOne has exactly the same data as NodeTwo because the data in NodeTwo is deleted before NodeOne is written to it.

Data in NodeOne			Data in NodeTwo		
Product (Primary key)	**Date (Primary key)**	**Qty**	**Product (Primary key)**	**Date (Primary key)**	**Qty**
Vanilla	Jan01	200	Vanilla	Jun02	375
Chocolate	Feb14	350	Almond	May24	190
Vanilla	Feb02	175	Chocolate	Apr14	89
Chocolate	Jan04	95	Cherry	Jun03	185
Cherry	Jun03	92			

Caution

The Delete/Write option should be turned off when a Data Editor node, a Strategic Network Optimization node, or a Production Scheduling - Process node is connected to a Data Object node in a data flow, or you may lose all your data.

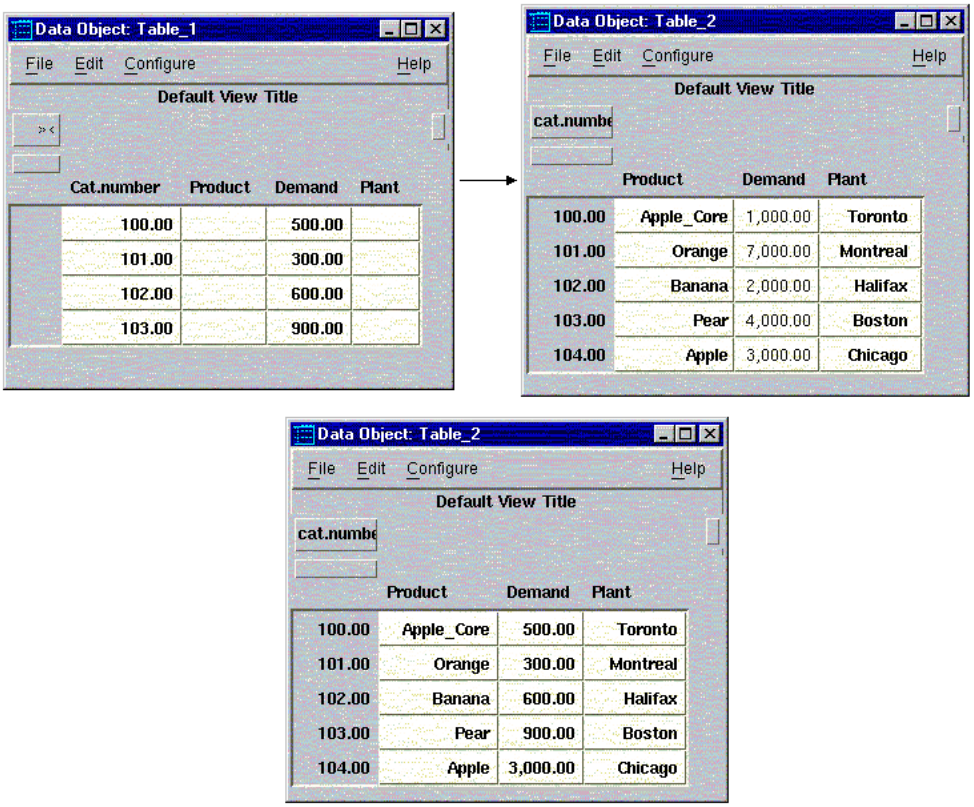
In the example below, the output data is deleted before it is used. The Data Editor writes the data out, replacing whatever data was in the table. The flow controller then deletes all of the contents of the table, losing all the output from the Data Editor node.



Example: Delete/Write option

- Changes Only option. You use the Changes Only option when you have fields that you want to update and you also have fields that you want to remain at their current values. For fields to remain unchanged, they must have a Null value in the corresponding fields of the input table.

  For example, in the case below, the Demand fields in Table 2 replace the Demand fields from Table 1 while all other fields remain unchanged.



Example:Changes Only option

- Delete Matched option. If primary key records in the output table match records in the input table, these records are deleted from the output table.
- Ignore Criteria option. If the Ignore Criteria option is selected, then the selection criteria is ignored. This is useful if you want to compare data that reflects the Selection Criteria with data that ignores the Selection Criteria.
- Selection Criteria option. With selection criteria, you can specify the records that you require from a data table during read or write operations. The intention is that if an relational database management system is used, the selection criteria can be pushed back to the database instead of being selected in memory. In this case, when reading data, only the data that is selected is added into memory.

  When writing to a data object node with selection criteria, only the records that satisfy the selection criteria are written to the data table through that data object node. Any records that fail to satisfy the selection criteria are returned through the exception output attach point of the data object node.

  For example, if you only want data for the product APPLE24 in the following table, you can enter the applicable selection criteria of $1.PRODUCT="APPLE24", and only the last three rows of the table are used:

Demand	Period	Product	Storage Location
1.400000	"Sep-05"	"APPLE12"	"BL-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"LA-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"MN-Apple-12"
4.400000	"Sep-05"	"APPLE12"	"PL-Apple-12"
17.600000	"Sep-05"	"APPLE24"	"AT-Apple-24"
8.800000	"Sep-05"	"APPLE24"	"BL-Apple-24"
17.600000	"Sep-05"	"APPLE24"	"LA-Apple-24"

  The expressions that you create and enter in the Expressions field are the criteria used in selecting records from the data table. You can use the following functions and operations in your expressions: AND ISNULL MAKEDATE MAKESYSKEY NOT OR XOR + - / * = () > >= < <= !=

  If you compare a date field to an actual date, you must use the MAKEDATE function. For example, to compare January 1, 2005, to a date field, enter:

```
$1.date1=MAKEDATE(2005,01,01)
```

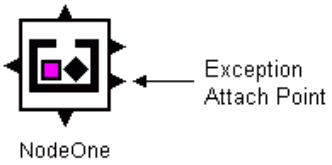
  If you want to compare two date fields, enter:

```
$1.date1=$1.date2
```

  You can also use data flow parameters in the selection criteria.

Understanding Handling Exception Output from Data Object Nodes

An exception output is the bottom right attach point fixed to the Data Object node icon.

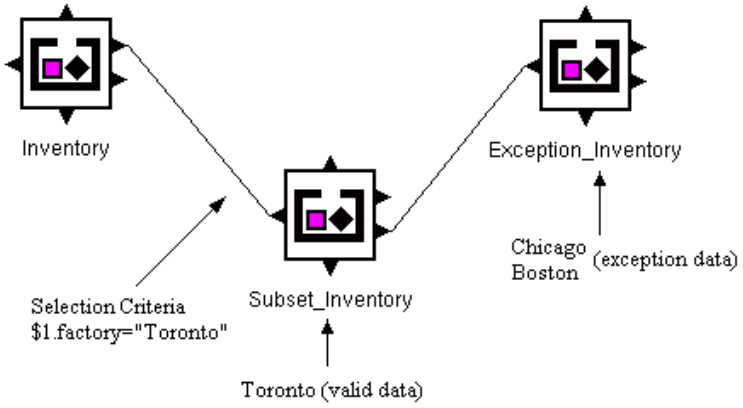


Exception output from a data object node

All data with the specified selection criteria is filtered before it is used in a data flow operation. Data that matches the selection criteria is called valid data.

Data that does not satisfy the selection criteria during a write is called exception data. The exception data passes to a Data Object node through the exception attach point.

The following example shows a Data Object node with selection criteria that limit the data to factories in Toronto:



Data object node selection criteria

In this example:

- The inventory data object node contains data for factories in Chicago, Boston, and Toronto.
- The valid data, which is the data associated with the factories in Toronto, is written to the Subset_Inventory Data Object node.
- The exception data is passed on to the Exception_Inventory Data Object node, which is connected to the exception attach point.

If the Subset_Inventory table contains records for factories other than Toronto, it is filtered out during the read. However, these original records are not deleted, assuming that the Delete/Write option is not set.

Understanding Viewing and Editing Data in Data Object Nodes

You can use the Data Editor to browse or edit the data represented by a data object node. The Data Editor displays the data in a spreadsheet format. You can view and edit the data and produce reports. If the Editable option is selected in the properties window for the data object node, you can save the changes to the data table.

If the Data Editor cannot be opened and an error message appears, ensure that the following items are correct:

- The data object is defined.
- The data table exists in the correct location.
- You have access rights to the data table.
- The data in the data table is not corrupt or incorrect, and the data matches the defined data types.

If the primary key is not unique, the Data Editor does not allow you to edit the table.

Windows Used to Set up Data Object Nodes

Window Name	Navigation	Usage
Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit.	View and edit nodes in data flows.
Properties	Access the Data Flow window. Do one of the following: • Select the data object node and select Edit, Properties. • Select the data object node, right-click and select Properties from the menu. • Double-click the node. **Note** Because of changes in Write Mode options, do not edit new Advanced Planning Agent projects in previous versions of Advanced Planning Agent.	View and edit data object node properties.
Browse Data Object	• Select the data object node. Select View, Browse Definition. • Select the data object node, right-click, and select Browse Definition from the shortcut menu.	View data object node definitions.

Window Name	Navigation	Usage
Data Object	• Select the data object node. Select View, Browse Data. • Select the data object node. Right-click and select Browse Data from the shortcut menu.	View or edit data in data object nodes.
Data Node Selection Criteria	Do one of the following: • Select Edit, the data object node, Properties. • Select the node, right-click and select Properties from the menu. • Double-click the node. Click Selection Criteria.	View or edit selection criteria.

Viewing and Editing Selection Criteria Fields and Functions

Access the Data Node Select Criteria window.

To view selection criteria fields and functions:

1. Click Fields.
2. Do one of the following:
   - To view fields in the input data object, click $1 Fields.
   - To view the $1 prefix, click Prefix Field List.
   - To view functions that you can use in selection criteria, click Functions.

Setting up Selection Filter Nodes

This chapter provides overviews of selection filter nodes, the union feature in selection filter nodes, selecting data from input data objects, lists the windows used to view and manage selection filter nodes, and discusses how to set up selection filter nodes.

Understanding Selection Filter Nodes

A selection filter selects the data that you want from a data table and filters out information that is not required.

A selection filter has the following features:

- Uses one or more input data objects, and one output data object.
- Uses expressions to select records of data from input data objects.
- Outputs only the selected records of data to the output data object. You specify which fields are output, and you can use expressions to perform calculations on the data before it is output.

You use output expressions to specify the data that is output to the fields in the output data object. You can use the expressions to perform calculations on the data. You create and edit the output expressions in the Output Expressions window.

Each FieldName has a corresponding expression that either outputs a value without changing it or changes the value in some way.

If the format of the output data records is the same as the input, then the output expression list can be left empty. In this case, the output data set references the selected records of the input data without creating new records. This process speeds up performance.

For example, the following input data table, represented by a data object node in a data flow, contains period, inventory, demand, and safety stock information:

Period	Inventory	Demand	Safety Stock
August 05	Red	25	3
September 05	Red	15	3
October 05	Blue	2	0

The output data table has the fields INVENTORY, PERIOD, and DEMAND.

If you want the period and demand information for the inventory Red and you don't want the safety stock values, you can use a selection filter in the data flow.

To get this information, you can use a selection filter with the selection criteria *$1.Inventory="Red"* for the input data object.

This expression selects only the records of data that contain the inventory Red.

After you select the required data, you can determine how the data is sent to the output data object. The output expressions for the output data object are as shown in the following table:

Field Name	Expression
INVENTORY	*$1.Inventory*
PERIOD	*$1.Period*
DEMAND	*$1.Demand*2*

The information from the Inventory field of the input data object (*$1.Inventory*) is sent to the INVENTORY field in the output data object. The information from the Period field of the input data object is sent to the PERIOD field of the output data object. The information from the Demand field of the input data object is multiplied by two and then sent to the DEMAND field in the output data object.

The resulting output data table is as follows:

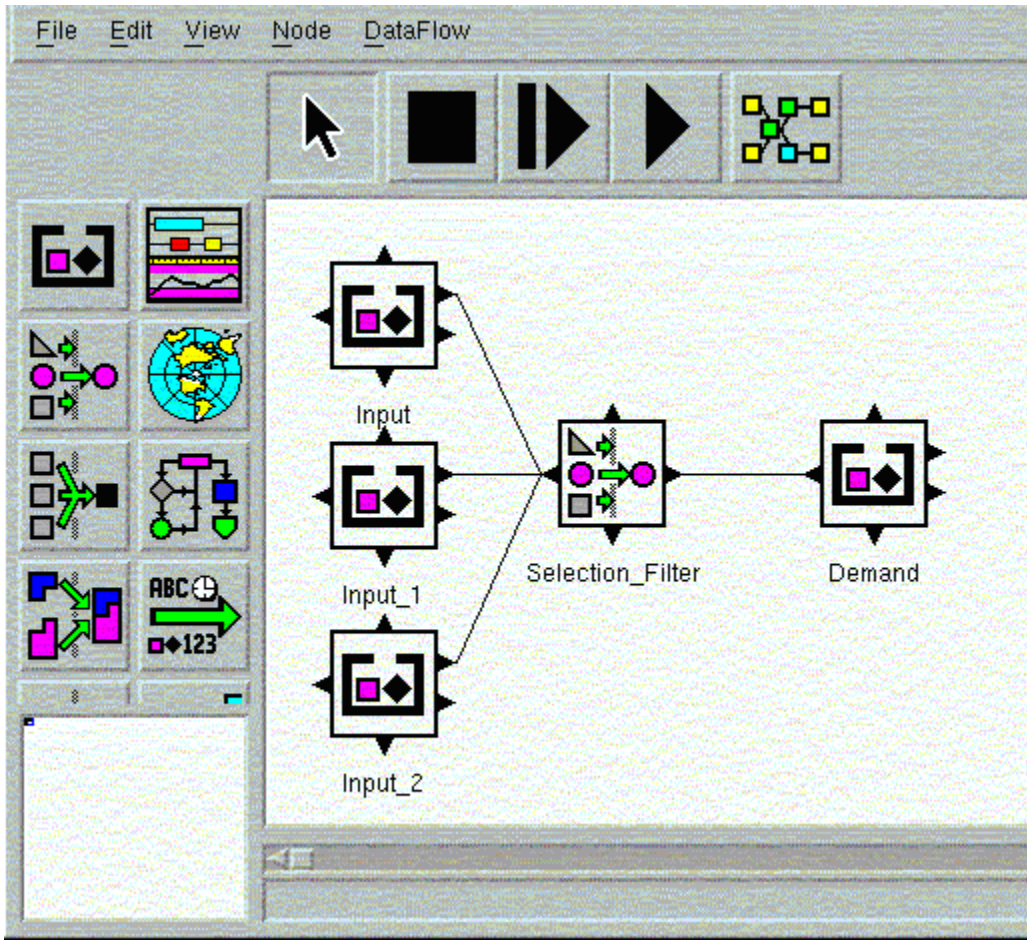
Inventory	Period	Demand
Red	August 05	50
Red	September 05	30

Understanding the Union Feature in Selection Filter Nodes

You can use the selection filter node to select records from several input nodes and output the united data to one or more data object nodes. To use the union feature, the data tables must have identical definitions. The data tables have identical definitions if they have the following properties:

- Same number of fields
- Same field names
- Equivalent data types
- Fields in the same order

The following data flow is an example of using the union feature:



Selection filter node

Understanding Selecting Data from Input Data Objects

In the Input Selection Criteria window, use expressions to determine which records are selected from the input data object. The expressions that you enter are the criteria used in selecting records from the input data object. A record is selected only if all of the input selection criteria are met. All the expressions must be true for a record to be selected.

You use filters along with expressions to manipulate the data used in a data flow. The kind of filter determines what kind of manipulation occurs, and the expressions specify the data used and how it is manipulated.

A filter, together with its expressions, selects the data that you want from an input record set, manipulates the data, and outputs part or all the data. In the following example, although the input is a data table in memory and the output is a data table in memory, you do not always require a data object node on both sides of a filter node. A filter node can be connected to another filter node.

You can use the following kinds of filters:

Filter	Description
Selection	Selects the data that you want
Aggregation	Groups the data from input records into one record and selects the aggregated data that you want
Join	Selects the data that you want from two data tables and joins the data according to criteria that you specify

An expression can be made up of the following components:

Expression Component	Example	Explanation
Number	400	Use any number.
String	"DC-1"	Use double quotation marks around strings.
Data set identifier and field name	$1.Demand	Use the syntax *$ x . fieldname* The data set identifier identifies the input attach point, the attached data set attached is the 2, 2, next, and so on. The *fieldname* is the name of a field in the input data set that is identified by the table identifier.
Symbol	+ - * / ()	Addition, subtraction, multiplication, division; use round brackets to specify order of operations.
Function	ABS($1.Demand)	Many functions are available for your use.
Nested expression	SUM(ABS($1.Demand))	You can nest functions and expressions.

The following are examples of valid expressions:

```
$1.Demand*100

NOW() + MINUTEDUR(5)

COS($2.Value)
```

You can use expressions with comparison symbols in any selection criteria window. The following are examples:

```
$1.Demand>500

$1.Color="Red" AND $1.Family="NoGloss"

MONTH($2.Period)=1
```

The following table shows how you can indicate equalities and inequalities, and how to use boolean operators *exp1* and *exp2* represent expressions:

Expression	Description
Exp1 = exp2	Returns true if the expressions are equal. Otherwise, returns false.

Expression	Description
Exp1 != exp2	Returns true if the expressions are not equal. Otherwise, returns false.
Exp1 < exp2	Returns true if *exp1* is less than *exp2* . Otherwise, returns false.
Exp1 > exp2	Returns true if *exp1* is greater than *exp2* . Otherwise, returns false.
Exp1 <= exp2	Returns true if *exp1* is less than or equal to *exp2* . Otherwise, returns false.
Exp1 >= exp2	Returns true if *exp1* is greater than or equal to *exp2* . Otherwise, returns false.
Exp1 AND exp2	Returns true if both expressions are non-zero or true. Otherwise, returns false.
Exp1 OR exp2	Returns true if one or both expressions are non-zero or true. Otherwise, returns false.
Exp1 XOR exp2	Returns true if exactly one expression is non-zero or true. Otherwise, returns false.

You can use many different functions to create your expressions. Each function has a specific syntax.

The functions include arithmetic functions, boolean functions, date functions, and others. Most functions have parameters, indicated within the round brackets.

The *x*, *y*, or other characters (such as *tol* and *numHours*) within the round brackets of a function frequently represent numbers. You can specify a field name if its value is a number or an expression that results in a number.

The *a,b,...* represents a list of numbers. You can specify a single field name if its values are numbers, a list of field names if they all contain numbers, or a list of expressions that result in numbers.

The following general syntax rules must be followed in all functions and expressions:

- Every function must be followed by a left and right parenthesis. For some functions the parentheses remain empty-for example, TRUE() and PI(). For other functions, the parentheses must contain arguments-for example, SUM($1.Prd1,$1.Prd2).
- Variables in functions are indicated by *x* , *y* , or some other variable name. In place of the variable name, you can enter:
  - A number
  - A field name, if the field contains a number
  - A function or expression that results in a number
- Variable lists in functions are indicated by a,b... . In place of this, you can enter:
  - A list of numbers
  - A field name if the field contains numbers
  - A list of names of fields that contain numbers
  - A list of functions or expressions that result in numbers

- Multiple functions and nested functions can be used in an expression.
- Do not mismatch data types. For example, a string function can use only string values, and a date function cannot use integers. Whether you use logical statements, arithmetic calculations, or date arithmetic, use only compatible data types in an expression.

  For example, functions and operations operate on types. Some functions work with multiple types. For example, *MIN(A,B,C)* works for any type as long as *A,B,C* are compatible data types, such as all strings.

Windows Used to Set up Selection Filter Nodes

Window Name	Navigation	Usage
Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit.	View and edit nodes in data flows.
Selection Filter	Access the Data Flow window. Do one of the following: • Select the selection filter node and select Edit, Properties. • Select the selection filter node, right-click and select Properties from the menu. • Double-click the node. Select a data flow and click Edit.	View and edit node properties.
Input Selection Criteria	Do one of the following: • Select the selection filter node and select Edit, Properties. • Select the selection filter node, right-click and select Properties from the menu. • Double-click the node. Click Selection Criteria.	Viewing and editing selection criteria expressions.
Output Expressions	Do one of the following: • Select the selection filter node and select Edit, Properties. • Select the selection filter node, right-click and select Properties from the menu. • Double-click the node. Click Output Expressions.	*Viewing and editing output expressions.*
Field and Function	In Expressions window, click Fields. **Note** The Field and Function window is common to all of the filter nodes.	

Setting up Selection Filter Nodes

Access the Selection Filter window.

To set up a selection filter node:

1. Create a selection filter node.
2. Connect the node to an input data object node and an output data object node.

   You can also connect the node to another filter node.

3. Select the node, right-click, and select Properties from the shortcut menu.
4. Enter a name for the node in the Selection Filter window.
5. Ensure that the names of the Input Data Object and Output Data Object are correct.
6. Click Selection Criteria.
7. Enter selection criteria expressions and click OK.
8. Click Output Expressions.
9. Enter output expressions and click OK.
10. (Optional) Enter Tcl scripts for the node.

Selecting Fields and Functions

Access the Field and Function window.

To select fields and functions, click one or more of the buttons described in the following table:

Button	Description
OK	Displays selected items in the Output Expressions window or the Selection Criteria window. Starts at the currently selected cell and overwrites the data. Closes the Field and Functions window.
Replace	Displays selected items in the Output Expressions window or the Selection Criteria window starting with the currently selected cell. Each selected item appears on a separate line and overwrites the data.
Select All	Selects all of the items in the list.
Clear	Deselects all of the items in the list.
Close	Closes the window.
Help	Displays on-line help.
$1 Fields	Displays a list of fields that are feeding into the first input attach point.
$2 Fields	Displays a list of fields that are feeding into the second input attach point.

Button	Description
Output Fields	Displays the field names from a node connected to the output attach point. If this node is a data object node, it displays the fields of the data table. If it is a filter node, it displays the field names of the output expression.
Functions	Displays a list of functions that you can use in expressions.
Prefix Field List	Displays the $1 or $2 prefix.

Setting up Aggregation Filter Nodes

This chapter provides overviews of aggregation filter nodes, determining data that is selected from input data objects, determining data that is grouped and sorted, determining aggregated data that is selected, determining data that is output, and bucketizing data in aggregation filter nodes, lists the windows that are used to set up aggregation filter nodes, and describes how to set up aggregation filter nodes.

Understanding Aggregation Filter Nodes

An aggregation filter groups the data from many different records of an input data object for each group. It creates an output record based on the group's properties and filters out information that is not required.

An aggregation filter can also sort and summarize information using rolling function sorts. Rolling function sorts can do the following:

- Group together records that have the same computed properties into major groups and subgroups. The subgroups can be sorted alphabetically (for characters), chronologically (for dates), and numerically (for numbers).
- A record is output for each subgroup, based on the properties of the current subgroup and the previous subgroups within the current major group. An example is a rolling 4-month average of customer demand.

An aggregation filter has the following features:

- Uses one input data object and one output data object.
- Uses expressions to select the records of data from the input data object that contains the data that you need.
- Groups and sorts the data.
- Aggregates and summarizes fields based on groups and criteria that you specify.
- Uses expressions to select the records of data from the aggregated data table that contains the data that you need.
- Outputs only the selected records of data to the output data object. You specify which fields are output, and you can use expressions to perform calculations on the data before it is output.

Determining Data that is Selected from Input Data Objects

In the Input Selection Criteria window for aggregation filter nodes, you use expressions to determine the records that are selected from input data objects.

The expressions entered in this window are the criteria used in selecting records of data from the input data object. A record is selected only if all of the input selection criteria are met. All the expressions must be true for a record to be selected.

Determining Data that is Grouped and Sorted

You use expressions to determine the fields that are aggregated, sorted, and grouped. You create and edit the expressions in the Aggregation Criteria window. This section discusses:

Grouping Information

If you specify expressions only in the Reset Sort Keys field, the expressions specify the records that are aggregated. The aggregation takes place only on the data that was selected by the input selection criteria. The selected input records are grouped if the values of the expressions are the same. The aggregation functions, such as *SUM*, compute their values for these groups.

Performing a Rolling Function Sort in the Aggregation Filter

If you specify expressions in both the Reset Sort Keys field and the Rolling Sort Keys field, a rolling function sort can be performed.

The following example illustrates a common error when using the rolling sort function with multiple records that have the same rolling and reset key values. For example, consider the following data table:

Plant	Item	Period	Production	Demand
A	A	Jan	10	10
A	A	Jan	14	14
A	A	Jan	15	12
A	A	Feb	10	11
A	A	Mar	15	13

Connect the table with an Aggregation filter. Set the following aggregation criteria:

```
Reset Sort Keys: $1.Plant and $1.Item
Rolling Sort Keys: $1.Period
```

For output, enter the expressions listed in the following table:

Field Name	Expression
Plant	$1.Plant
Item	$1.Item
Period	$1.Period

Field Name	Expression
Month_demand_0	SUM($1.Demand)
Month_demand_1	R_SUM($1.Demand, 2)
Month_demand_2	R_SUM(SUM($1.Demand), 2)

After running this flow, the results are:

Plant	Item	Period	Month_demand_0	Month_demand_1	Month_demand_2
A	A	Jan	36	14 or 10 or 12	36
A	A	Feb	11	25 or 21 or 23	47
A	A	Mar	13	24	24

In the output table, the record corresponding to the periods Jan and Feb returns random values for the column Month_Demand_1 because $1.Demand is the first argument of R_SUM function.

To avoid random results, apply one of the functions SUM, MIN, MAX, or AVG to $1.Demand.

Using Reset Sort Keys

The Reset Sort Keys specify expressions so that the records of data that have the same information, as specified by the expressions, are grouped together. The information in those records can be summarized and updated with each aggregated record. Each group of records can be summarized by a rolling summary.

Using Rolling Sort Keys

The Rolling Sort Keys specify the fields on which you want to sort the records within the aggregated groups. Sort data in either ascending or descending order. The field on the right side can be toggled between displaying a + plus or - minus sign. Click on it to change from one sign to another.

If you perform a rolling function sort, you can use the following functions in the output expressions:

Function	Description
R_AVG	Calculates the rolling average.
R_COVER	Calculates the periods of coverage for an inventory level.
R_MAX	Finds the rolling maximum value.
R_MIN	Finds the rolling minimum value.
R_PREV	Uses the previous value. If no previous value exists, the default is used.
R_PRODUCT	Calculates the rolling product.
R_SUM	Calculates the rolling sum.

Determining Aggregated Data that is Selected

You use expressions to determine the records of aggregated data that are selected. You create and edit the expressions in the Output Selection Criteria window.

The Output Selection Criteria window is similar to the Input Selection Criteria window except that the expressions apply to the output of the aggregation, and you can use aggregation functions and rolling functions.

Determining Data that is Output

You use output expressions to specify the data that is output to the fields in the output data object. You can use the expressions to perform calculations on the data. You create and edit the output expressions in the Output Expressions window.

Example: Using an Aggregation Filter

The following input data object, represented by a data object node in a data flow, contains family, SKU, period, and demand information:

No.	Family	SKU	Period	Demand
1	Blue	B1	1	6
2	Red	R1	1	3
3	Yellow	Y3	2	4
4	Blue	B1	2	5
5	Blue	B2	1	4
6	Blue	B1	2	3

The output data table has fields FAMILY, PERIOD, and DEMAND.

If you want the demand data by family for Blue in Period 1, and the data by SKU and for other colors is not needed, you can use an aggregation filter in the data flow.

To get the input information, you can use an aggregation filter with the selection criteria *$1.Family="Blue"* for the input data object. This expression selects only the records of data that contain the family Blue. The selected input data is as follows:

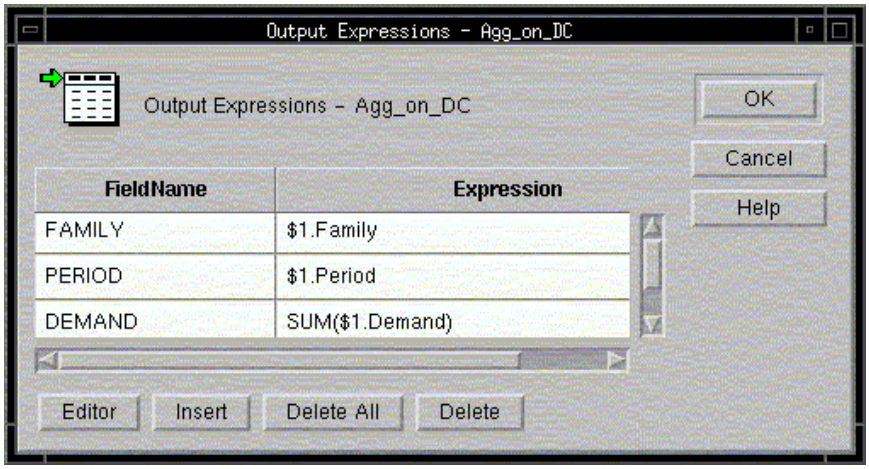
No.	Family	SKU	Period	Demand
1	Blue	B1	1	6
4	Blue	B1	2	5
5	Blue	B2	1	4
6	Blue	B1	2	3

To aggregate this information, enter the expression *$1.Period* as the aggregation criteria in the Reset Sort Keys field. This expression aggregates the information by period. The records are grouped by period. The grouped data is stored temporarily in the following format:

Group	No.	Family	SKU	Period	Demand
1	1	Blue	B1	1	6
1	5	Blue	B2		4
2	4	Blue	B1	2	5
2	6	Blue	B1		3

Because you only want information for Period 1, you enter the expression *$1.Period="1"* as the output selection criteria. This expression selects only the records of aggregated data that contain the information for Period 1.

Having selected and aggregated the required data, you can determine how the data is to be sent to the output data object. The output expressions for the output data object are shown in the following screenshot:



Output data object expressions

The information from the Family field of the input data object (*$1.Family*) is sent to the FAMILY field in the output data object. The information from the Period field of the input data object is sent to the PERIOD field of the output data object. The values in the Demand field of the input data object for each aggregated group are added together and then sent to the DEMAND field in the output data object.

The resulting output data table is as follows:

Family	Period	Demand
Blue	1	10

Example: Aggregation Filters Doing Rolling Function Sorts

Consider a data table that contains the following data about period, product, and cost:

Period	Product	Cost
4	Yellow	5
1	Purple	2.5
1	Purple	8.5
2	Purple	7.5
2	Purple	1
3	Yellow	15
4	Purple	6.5
4	Purple	8
4	Purple	12
1	Blue	22
1	Blue	80
2	Blue	250
3	Purple	25
3	Blue	20
3	Blue	50
3	Blue	70

You can use a rolling sort to find the total cost for each product, by period.

Specify the Product field as the reset sort key. This process means that each time that the product changes, the function which calculates the rolling cost is reset.

Specify the Period field as the rolling sort key. Using the *R_SUM* function for the Cost field, the rolling cost for each period by product is calculated.

The output expressions might be as follows:

Field	Expression
Product	*$1.Product*
Period	*$1.Period*
Rolling Cost	*R_SUM(SUM($1.Cost),3)*

In R_SUM expression, the second parameter, three in this example, indicates that the *R_SUM* remembers only the last three values.

The result is as follows:

Product	Period	Rolling Cost
Blue	1	102
Blue	2	352
Blue	3	492
Purple	1	11
Purple	2	19.5
Purple	3	44.5
Purple	4	60
Yellow	3	15
Yellow	4	20

Notice the following results:

- The records are grouped by product. This grouping is done by the reset sort.
- The periods are sorted within each product group. This sorting is done by the rolling sort.
- The rolling cost for each period is calculated for each product. This calculation is done by the output expression *R_SUM(SUM($1.Cost),3)* .

The following example illustrates a common error when using the rolling sort function with multiple records that have the same rolling and reset keys. Consider the following data table:

Plant	Item Date	Date	Product	Demand
A	A	Jan	10	10
A	A	Jan	15	12
A	A	Feb	10	11
A	A	Mar	15	13

Connect the table with an Aggregation filter. Set the following aggregation criteria:

```
Reset Sort Keys: $1.Plant and $1.Item

Rolling Sort Keys: $1.Period
```

For output, enter the expressions listed in the following table:

Field Name	Expression
Plant	$1.Plant
Item	$1.Item

Field Name	Expression
Period	$1.Period
Month_demand_0	SUM($1.Demand)
Month_demand_1	R_SUM($1.Demand, 2)
Month_demand_2	R_SUM(SUM($1.Demand), 2)

After running this flow, the results are:

Plant	Item	Period	Month_demand_0	Month_demand_1	Month_demand_2
A	A	Jan	36	14 or 10 or 12	36
A	A	Feb	11	25 or 21 or 23	47
A	A	Mar	13	24	24

In the output table, the record corresponding to the periods Jan and Feb returns random values for the column Month_Demand_1 because $1.Demand is the first argument of R_SUM function.

To avoid random results, apply one of the functions SUM, MIN, MAX, or AVG to $1.Demand.

Bucketizing Data in Aggregation Filter Nodes

Bucketizing is available in aggregation filter nodes. This feature combines events and the dates when those events occur and groups them into time buckets.

Once you bucketize the data, you can then perform output functions in the Output Expressions window.

This section dicusses the bucketvalues, bucketlengths, and bucketperiods methods of bucketizing in aggregation filter nodes.

Bucketvalues Method

This method specifies the exact boundaries of the buckets:

BUCKETVALUES (*$1.eventField, bucket1Start, bucket2Start,... bucketEnd)*

where *$1.eventField* is the event date field in the input data object, *bucket1Start* , *bucket2Start* are the start dates for all the buckets in the horizon. The *bucketEnd* is the end of the last bucket in the horizon.

Bucketlengths Method

This method has a start date and a length of time, such as, a start date of January 1 plus 30 days. Enter the following:

BUCKETLENGTHS (*$1.eventField, startValue, bucketLength, bucket2Length,...*)

where *$1.eventField* is the event date field in the input data object, followed by the *startValue* of the first bucket.

Bucketperiods Method

This method only bucketizes increments of time. Enter the following:

BUCKETPERIODS (*$1.dateField, "f"/"b", startDate, bucketLength, #buckets,...*)

where *$1.dateField* is the event date field in the input data object, s *tartDate* is the start date of the horizon, and *bucketLength/#buckets* are a series of pairs that specify the size of the bucket and the number of buckets in a row of that size. The *bucketLength* is either *"hour" "day" "month" "year"*, or an exact duration in *seconds.* The time measurements must be in quotes.

Bucketizing Examples

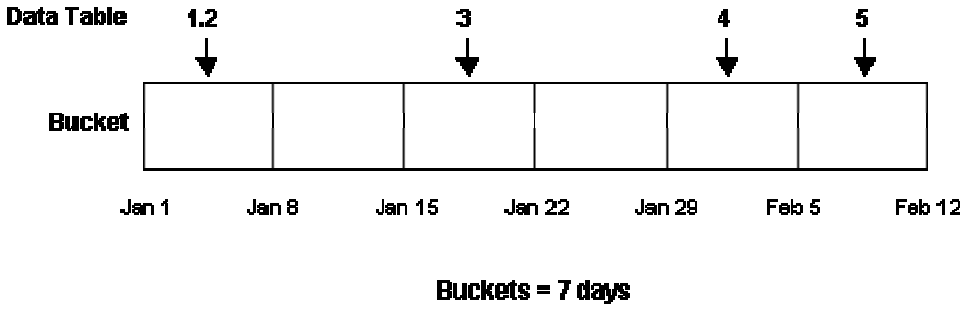
Consider the following data table for each of the following bucketizing examples:

Table/Number	Date	Production/Value
1	Jan 1	1000
2	Jan 5	500
3	Jan 20	1500
4	Jan 30	800
5	Feb 8	1200

BUCKETVALUES Example # 1

```
BUCKETVALUES($1.Date,MAKEDATE(1997,1,1),MAKEDATE(1997,1,8),MAKEDATE(1997,1
,15),MAKEDATE(1997,1,22),MAKEDATE(1997,1,29),MAKEDATE(1997,2,5),MAKEDATE(1
997,2,12))
```

The following graphic illustrates the result of this expression:

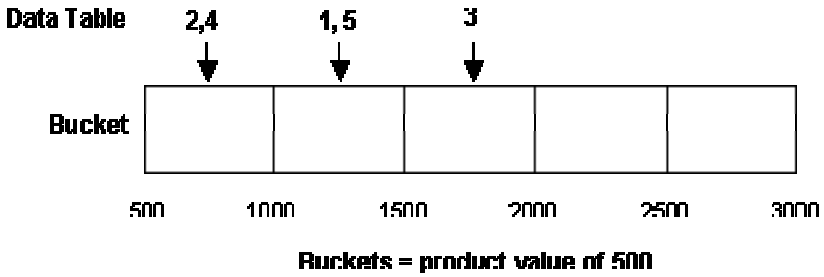


Bucketvalues example 1

BUCKETVALUES Example # 2

```
BUCKETVALUES($1.ProductionValue,500,1000,1500,2000,2500,3000)
```

The following graphic illustrates the result of this expression:

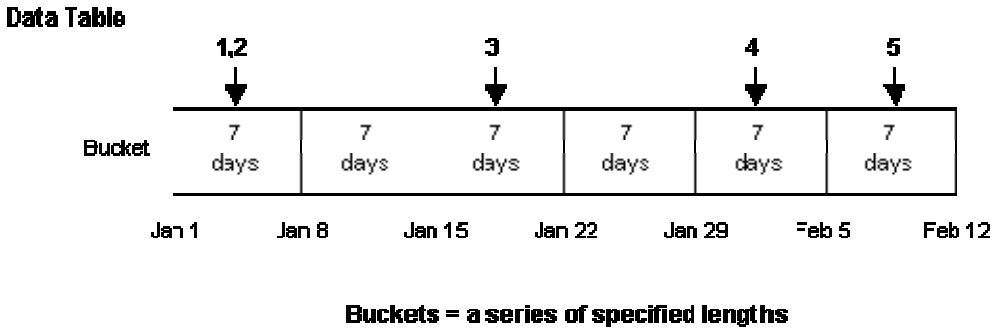


Bucketvalues example 2

BUCKETLENGTHS Example

```
BUCKETLENGTHS($1.Date,MAKEDATE(2005,1,1),DAYDUR(7),DAYDUR(7),DAYDUR(7),
DAYDUR(7),DAYDUR(7),DAYDUR(7))
```

The following graphic illustrates the result of this expression:

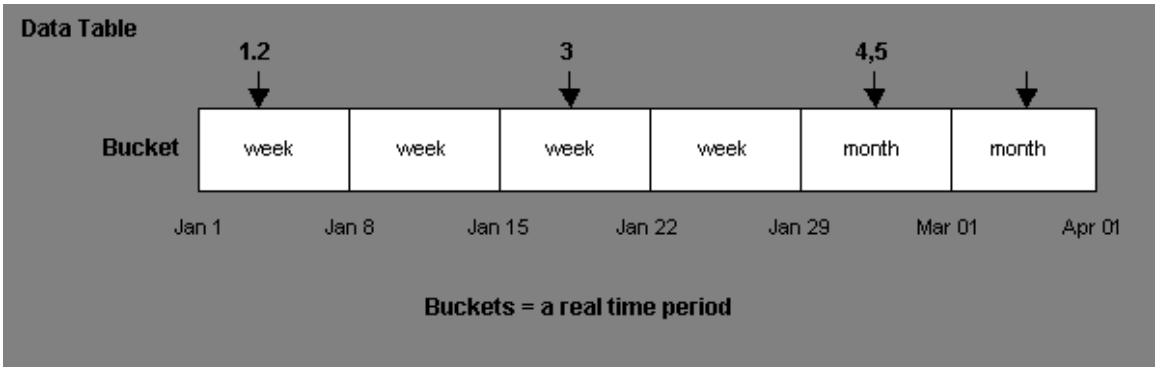


Bucketlengths example

BUCKETPERIODS Example

```
BUCKETPERIODS($1.Date,"f",MAKEDATE(2005,1,1),"week",4,"month",2 )
```

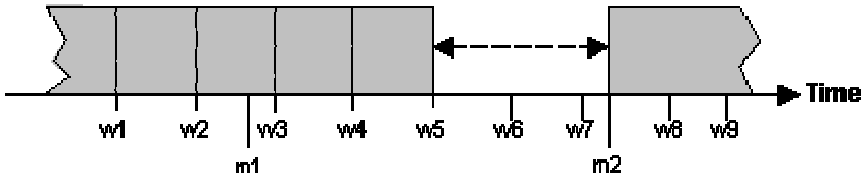
The following graphic illustrates the result of this expression:



Bucketperiods example

Handling Gaps and Overlaps in Horizons When You Bucketize

Occasionally, a gap can exist between the end of one series of buckets and the beginning of the next series of buckets. For example, if you specify four weekly buckets beginning at w1, followed by monthly buckets, the bucketizer places the first monthly bucket at m2, leaving a gap between w5 and m2. You cannot have gaps in a horizon because the bucketizing feature automatically inserts extra weekly buckets until it meets or passes the next monthly boundary, in this case to w8. If the end of the last weekly bucket exactly lines up with the start of the first monthly bucket, then that line becomes the bucket boundary.

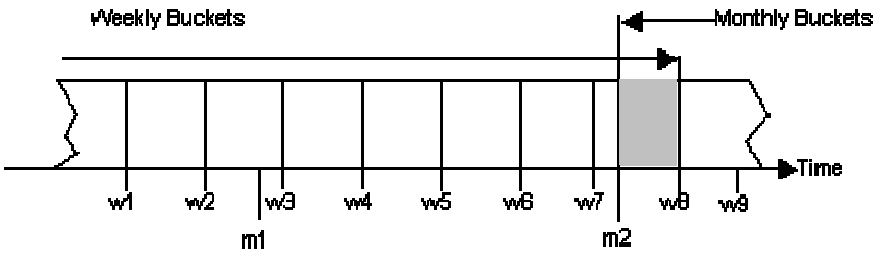


Handling gaps in horizons when you bucketize

If an overlap exists (as in this example, where w8 is past m2.) then it is handled using the method described in the following section.

Since the boundaries of weekly buckets may not always line up with the boundaries of monthly or yearly buckets, either the end of one bucket or the beginning of the next must be truncated to ensure that no range is evaluated twice.

For example, in the following graphic, you are bucketizing with a series of weekly buckets that end at w8.



Handling overlaps in buckets when you bucketize

Alternately, the system may have inserted extra weekly buckets up to w8 as above), followed by a series of monthly buckets. However, w8 does not line up exactly with any monthly bucket boundary, so either the last weekly bucket must be truncated to m2 or the first monthly bucket must be truncated to w8. You determine which is used with the f/b option in BucketPeriods. If f is used, the bucketizer truncates the first monthly bucket forward to w8. If b is used it truncates the last weekly bucket back to m2.

Windows Used to Set up Aggregation Filter Nodes

Window Name	Navigation	Usage
Data Flow	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit.	View and edit nodes in data flows.
Aggregation Filter	Access the Data Flow window Do one of the following: • Select the aggregation filter node and select Edit, Properties. • Select the aggregation filter node, right-click and select Properties from the menu. • Double-click the node.	View and edit aggregation filter node properties.
Aggregation Criteria	Do one of the following: • Select the aggregation filter node and select Edit, Properties. • Select the aggregation filter node, right-click and select Properties from the menu. • Double-click the node. Click Aggregate On.	View and edit aggregation criteria.
Input Selection Criteria	Do one of the following: • Select the aggregation filter node and select Edit, Properties. • Select the aggregation filter node, right-click and select Properties from the menu. • Double-click the node. Click Selection Criteria.	View and edit input selection criteria.

Input Selection Criteria	Do one of the following: • Select the aggregation filter node and select Edit, Properties. • Select the aggregation filter node, right-click and select Properties from the menu. • Double-click the node. Click Selection Criteria in the Input area.	View and edit input selection criteria.
Output Selection Criteria	Do one of the following: • Select the aggregation filter node and select Edit, Properties. • Select the aggregation filter node, right-click and select Properties from the menu. • Double-click the node. Click Selection Criteria in the Output area.	View and edit output selection criteria.
Output Expressions	Do one of the following: • Select the aggregation filter node and select Edit, Properties. • Select the aggregation filter node, right-click and select Properties from the menu. • Double-click the node. Click Output Expressions.	View and edit output expressions.

Setting up Aggregation Filter Nodes

Access the Aggregation Filter window

To set up an aggregation filter node:

1. Create an aggregation filter node.
2. Connect the node to an input data object node and an output data object node. You can also connect to another filter.
3. Select the node, right-click, and from the menu, select Properties.
4. In the Aggregation Filter window, enter a name for the node in the Name field.

5. Ensure that the names of the Input Data Object and Output Data Object, which are filled in automatically, are correct.
6. Enter the input selection criteria, which determine the data that is selected from input and for output.
7. Click Aggregate On and enter the aggregation criteria, which determine the data that is aggregated and how the rolling sorts are performed.
8. Enter the output expressions, which determine what output data is selected and to what fields the data is sent. You can also perform calculations on the data.
9. (Optional) Enter Tcl scripts for the node.

Setting up Join Filter Nodes

This chapter provides overviews of join filter nodes, lists windows used to set up join filter nodes and discusses how to set up join filter nodes.

Understanding Join Filter Nodes

This section provides overviews of:

- Join filter nodes.
- Determining data that is selected from input data objects.
- Determining data that is joined.
- Determining joined data that is selected in join filter nodes
- Determining data that is output.
- Collecting data that is not joined in join filter nodes.

Join Filter Nodes

A join filter selects the data that you want from two data tables, joins the data according to criteria that you specify, and filters out information that is not required.

A join filter:

- Uses two input data objects and at least one output data object. It can have output data objects containing data that does not meet the join criteria.
- Uses expressions to select records of data from the input data objects.
- Matches data records according to specified criteria.
- Sends results computed from the selected records to the output data set. You specify which fields and computed values are output. You can also gather the information that does not meet the join criteria, which is called exception data.

The fields that the join filter outputs match the fields that are defined in the output data object.

Determining Data that is Selected From Input Data Objects

You use expressions to determine the records that are selected from the input data object. You create and edit the expressions in the First Input Selection Criteria window and the Second Input Selection Criteria window. These records are selected before the join takes place.

You use expressions to determine the records that are selected from the input data object. You create and edit the expressions in the First Input Selection Criteria window and the Second Input Selection Criteria window. These records are selected before the join takes place.

The expressions listed in this window are the criteria used in selecting records of data from the first input data object. This expression must not reference fields from the second input data set. A record is selected only if all of the input selection criteria are met. All of the expressions must be true for a record to be selected.

The expressions listed in the second input expression window are the criteria used in selecting records of data from the second input data object. These expressions must not reference fields from the first input data set. A record is selected only if all of the input selection criteria are met. All of the expressions must be true for a record to be selected.

Determining Data that is Joined

You use expression lists to determine the fields or records that must match before the records can be joined. You create and edit the expressions in the Join Criteria window. The first input expression list must not reference the second input data set, and the second input expression list must not reference the first input data set.

If the first input expression equals the second input expression, a join of the two records (one from each input data object) occurs.

The expressions entered in the Join Criteria window specify the fields that are joined. The join takes place on the data that was selected by the input selection criteria. Records are joined when the first input expression is equal to the second input expression and they pass the output selection criteria.

You can only use fields from the first data object in the First Input Expressions fields, and you can only use fields from the second data object in the Second Input Expressions fields.

Each line of the Expressions field can have an expression.

Determining Joined Data that is Selected in Join Filter Nodes

You use expressions to determine the records that are selected from the joined data. You create and edit the expressions in the Output Selection Criteria window. Expressions can mix references from data sets.

Determining Data that is Output

You use output expressions to specify the data that is output to the fields in the output data object. You can use the expressions to perform calculations on the data. Create and edit the output expressions in the Output Expressions window. Expressions can mix references to input data objects 1 and 2.

The Field Names are the names of the fields in the output data object. Each field has a corresponding expression that either outputs a value without changing it or changes the value in some way.

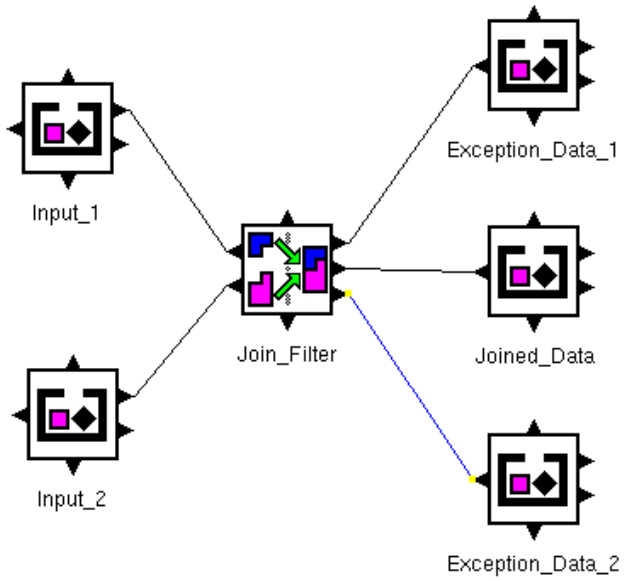
Collecting Data that is not Joined in Join Filter Nodes

With the join filter, you can collect the data that is not joined, which is called the exception data. Exception data for a input data object must have passed the input selection criteria and not satisfied the join criteria with any record from the other input data object. If it passed the join criteria, it did not

pass the output selection criteria for any record from the other input data object. The exception data for each input data object can be saved in an output data object.

If you want to collect exception data, set up the join node as follows:

- To the first output attach point on the join filter, connect a data object or a filter node to contain the exception data from the first input data object node.
- To the second output attach point on the join filter, connect the data object or filter node to contain the joined data.
- To the third output attach point on the join filter, connect a data object or filter node to contain the exception data from the second input data object node.



Collecting exception data from a join filter node

When you connect an exception data object node to a join filter, its name appears in the Exception Outputs section of the properties window for the join filter. You can connect one, both, or neither exception data object nodes. If you do not connect an exception data object node, its exception data is not collected.

Example: Join Filter Exceptions

Suppose there are two demand tables that contain the following data:

West Coast Demand			East Coast Demand		
Product	**Date**	**Qty. West**	**Product**	**Date**	**Qty. East**
1	Jan 1	2	1	Jan 1	4
2	Jan 3	3	2	Jan 3	7
1	Jan 5	8	2	Jan 9	12

In this example, the join filter joins on product and date. The table containing the join information is as follows:

Product	Date	Qty. West	Qty. East
1	Jan 1	2	4
2	Jan 3	3	7

The first join exception table consists of the data in the West Coast Demand table that did not match the join criteria, and the second table consists of the East Coast Demand that did not match the join criteria.

West Coast Demand Exceptions			East Coast Demand Exceptions		
Product	Date	Quantity	Product	Date	Quantity
1	Jan 5	8	2	Jan 9	12

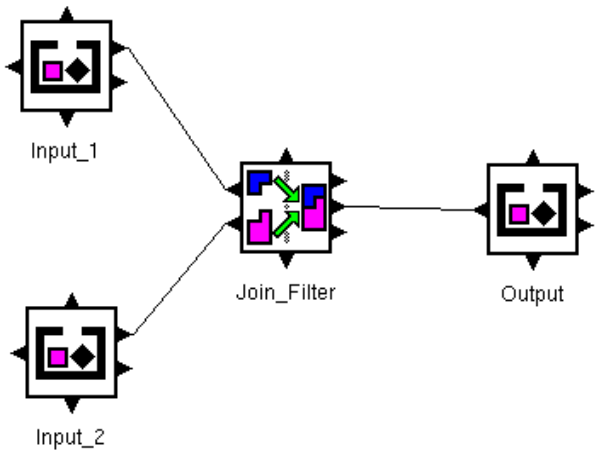
Example: A Join Filter Without Join Exceptions

The following two data tables are represented by data object nodes in a data flow. Table A contains the demand information by month on the family level. Table B contains the demand percentage values for each SKU on the family level and shown below:

Table A			Table B		
Family	Month	Demand	Family	SKU	Percentage of demand
Blue	January	50	Blue	B1	50
Red	September	50	Blue	B2	50
Purple	January	100	Purple	P1	20
Purple	September	100	Purple	P2	80
Red	January	50	Red	R1	90
			Red	R2	10

The output table has the fields SKU, MONTH, and SKUDEMAND.

If you want the demand information for each month by SKU for the families Blue and Red, you can use a join filter as shown in data flow below:



Example: Join filter

To get the demand information, you can use a join filter with the selection criteria *$1.Family="Blue" OR $1.Family="Red"* for the first input data object. Use the expression *$2.Family="Blue" OR $2.Family="Red"* for the second input data object.

Each expression selects only the records of data that contain the families Blue or Red from the input data objects. The selected input data is as follows:

Table A			Table B		
Family	**Month**	**Demand**	**Family**	**SKU**	**Percentage of demand**
Blue	January	50	Blue	B1	50
Red	September	50	Blue	B2	50
Red	January	50	Red	R1	90
			Red	R2	10

Having selected the required data, you can determine how the data is to be joined. In Join Criteria window, enter the expression *$1.Family* in the First Input Expressions column. Enter the expression *$2.Family* in the Second Input Expressions column. The Join Criteria window species that records in Table A are to be joined with records in Table B if the value of the Family field is the same. The resulting joined data is as follows:

Family	Month	SKU	Demand	Percentage of Demand
Blue	January	B1	50	50
Blue	January	B2	50	50
Red	September	R1	50	90
Red	September	R2	50	10
Red	January	R1	50	90

Family	Month	SKU	Demand	Percentage of Demand
Red	January	R2	50	10

Having joined the required data, you can determine how the data is to be sent to the output data object. The output expressions for the output data object are as follows:

Field Name	Expression
SKU	$2.SKU
MONTH	$1.Month
SKUDEMAND	$1.Demand*$2.PercentofDemand/100

The information from the SKU field of the second input data object (*$2.SKU*) is sent to the SKU field in the output data object. The information from the Month field of the first input data object is sent to the Month field of the output data object. The values in the Demand field of the first input data object are multiplied by the values in the PercentofDemand field in the second input data object and then sent to the SKUDEMAND field in the output data object.

The resulting output data table is as follows:

SKU	MONTH	SKUDEMAND
B1	January	25
B2	January	25
R1	September	45
R1	January	45
R2	September	5
R2	January	5

Note

The join criteria can be replaced by the output selection criteria. The difference is that the join runs much faster if you use the join criteria. Use the join criteria whenever possible, even if it is only for part of the join condition.

Window Used to Set up Join Filter Nodes

Window Name	Navigation	Usage
Join Filter	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Do one of the following: • Select the node and select Edit, Properties. • Select the node, right-click and select Properties from the menu. • Double-click the node.	View and edit join filter node properties.

Setting up Join Filter Nodes

Access the Join Filter window.

To set up a join filter node:

1. Create a join filter node.
2. Connect the node to two input data object nodes and an output data object node. You can also connect to another filter.
3. If required, connect the node to one or two additional output data objects that will contain the data that does not meet the join criteria.
4. Select the node, right-click and from the menu, select Properties.
5. In the Join Filter window, enter a name for the node in the Name field.
6. Ensure that the names of the Input Data Object and Output Data Object, which are filled in automatically, are correct. If used, also ensure that the Exception Outputs are correct.
7. Enter the input selection criteria expressions, which determine the data that is selected, for each input data object node. Enter the join criteria, which determine the data that is joined.
8. Enter the output expressions, which determine what output data is selected and to what fields the data is sent. You can also perform calculations on the data.
9. (Optional.) Enter Tcl scripts for the node before and after you import the data.

Setting up Production Scheduling - Process Nodes

This chapter provides an overview of Production Scheduling - Process nodes, lists the window used to set up Production Scheduling - Process nodes, and discusses how to set up Production Scheduling - Process nodes.

Understanding Production Scheduling - Process Nodes

You can use a Production Scheduling - Process node to integrate the Production Scheduling - Process system into a data flow. For example, you can use Advanced Planning Agent to input your starting inventory and demand information into the Production Scheduling - Process system. You can then create and export your schedule.

A Production Scheduling - Process node requires at least one input data object node and can have any number of output data object nodes. Each of these data object nodes has an associated smart definition file. The definition file enables the Production Scheduling - Process system to interpret the data that is in the data object node. It converts information into a format that the Production Scheduling - Process system can understand. Only data object nodes for flat file data tables can be connected to a Production Scheduling - Process node.

The smart definition file must be in the same format as the one used by the PSI Editor and the SMARTIN and SMARTOUT commands. The data tables represented by the data object nodes must contain fields defined in the smart definition files. You can create smart definition files in the PSI Editor of the Production Scheduling - Process system or with a text editor such as vi or emacs.

When a Production Scheduling - Process node is run, the data from the input data objects is imported into the Production Scheduling - Process system with the SMARTIN command, and the schedule is updated. If there are output data object nodes, the data is then exported with the SMARTOUT command. These commands are automatically executed when you run the data flow; you do not need to manually enter the SMARTIN and SMARTOUT commands.The underlying data table can contain the header created by the SMARTOUT command without affecting Advanced Planning Agent.

The following example shows how you can integrate the Production Scheduling - Process system into an Advanced Planning Agent data flow. Although the example is simplistic, the steps that you follow in a more realistic situation are basically the same, although there may be other required steps or complications.

The Production Scheduling - Process case is called EXAMPLE, and it has inventories such as BLUEMIX, REDMIX, YELLOWMIX, and so on. You want to find the total amount of each inventory produced for each week of the schedule and export that data to a data object in Advanced Planning Agent. Once the data is in Advanced Planning Agent, you want to be able to view it using the Data Editor and manipulate the data in some way.

To begin, you start the Production Scheduling - Process system and open the case EXAMPLE. In the PSI Editor, you set up a view of the data that you want to export. In this example:

- All the inventories are selected.

- The parameter for the inventories that are selected is Produced.
- The Times selected are Weeks from the start of the schedule to the end.

You can then set up directory paths for the definition and data files, and create a smart definition file.

Window Used to Set up Production Scheduling - Process Nodes

Window Name	Navigation	Usage
Production Scheduling – Process node	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Do one of the following: • Select the node and select Edit, Properties. • Select the node, right-click and select Properties from the menu. • Double-click the node.	Set up Production Scheduling - Process nodes.

Setting up Production Scheduling - Process Nodes

1. Create a Production Scheduling - Process node.
2. Connect the node to its input data object nodes and, optionally, its output data object nodes.
3. Select the Production Scheduling - Process node, right-click, and from the menu, select Properties.
4. In the Production Scheduling - Process window, enter a name for the node in the Name field.
5. Ensure that the names of the input and output data object nodes, which are filled in automatically, are correct.
6. In the Case File field, enter the path and name of the Production Scheduling - Process case.
7. Click Use GUI if you want to display the Production Scheduling - Process interface.

   When the GUI is displayed, the flow is stopped and you can use the Production Scheduling - Process system. For example, you can run an algorithm.
8. Specify the input and output definition files.
9. For the input definition file, select Input Data Objects and, in the Definition File column, enter the names of the definition files that correspond to each data object in the Definition File column.
10. For the output definition file, select Output Data Objects and, in the Definition File column, enter the names of the definition files that correspond to each data object in the Definition File column.

11. (Optional.) In the Production Scheduling - Process Procedures section, enter the Production Scheduling - Process procedures that you want to run before and after you import the data.
12. (Optional.) In the Tcl Scripts section, enter the Tcl scripts that you want to run before or after you import the data.

Creating and Managing Strategic Network Optimization Nodes

This chapter provides an overview of Strategic Network Optimization nodes, lists the window used to set up Strategic Network Optimization nodes, and discusses how to set up Strategic Network Optimization nodes.

Understanding Strategic Network Optimization Nodes

This section provides overviews of:

- Strategic Network Optimization nodes
- Specifying input and output files.
- Using Strategic Network Optimization nodes without launching the system.

Strategic Network Optimization Nodes

You can use the Strategic Network Optimization node to integrate the Strategic Network Optimization system into a data flow. Strategic Network Optimization enables you to solve a wide range of manufacturing, distribution, and logistics problems.

A Strategic Network Optimization node requires at least one input data object node and can have any number of output data object nodes. Each input data object node has an associated input .smart file and .view file. Each output data object node has an associated smart graph (.smt) and output .smart definition file.

The input .smart file must exist before the data flow is run. You can create a .smart file by applying a smart graph (.smt) to a model. You should use the .view file generated by the Strategic Network Optimization DISP executable.

When the data flow runs, the following events occur:

1. The data from an input data object node is indexed with its input .smart file, and a temporary import file is created.
2. The temporary import file is loaded into the Strategic Network Optimization system.
3. Optionally, the GUI is displayed.
4. Optionally, the Strategic Network Optimization model is solved.
5. The specified smart graphs are applied, and the results are stored in an output .smart definition file.
6. A copy of the output .smart definition file is stripped of its SNO-specific information, and the resulting data is output to the specified output data object node.

Note

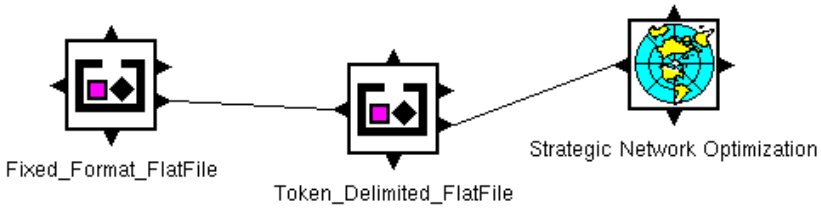
When running a data flow that includes a Strategic Network Optimization node, a log file called `linx.log` is generated in the current directory. This log records any errors that may occur.

Specifying Input and Output Files

Each input data object node has an associated input .smart file and .view file. Each output data object node has an associated smart graph (.smt) and output .smart definition file.

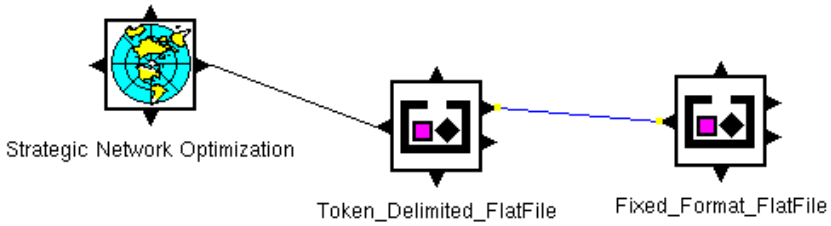
The input .smart file must exist before the data flow is run. You can create a .smart file by applying a smart graph (.smt) to a model.

Input to a Strategic Network Optimization node cannot come directly from a fixed format flat file. You must first direct the Strategic Network Optimization input to a token delimited flat file, as shown in the following graphic:



Input from flat files

Output from a Strategic Network Optimization node cannot go directly to a fixed format flat file. You must first direct the Strategic Network Optimization output to a token delimited flat file, then on to a fixed format file, as shown in the following graphic:



Output to flat files

Note

The Delete/Write option should be turned off when a Strategic Network Optimization node is connected to a Data Object node in a data flow, or you may lose all your data.

Using Strategic Network Optimization Nodes Without Launching the System

Strategic Network Optimization is not launched if the following conditions are met:

- The Use GUI button is off.
- The Import/Export Only button is on.
- No commands exist in the Commands window.

Window Used to Set up Strategic Network Optimization Nodes

Window Name	Navigation	Usage
Strategic Network Optimization node	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Do one of the following: • Select the node and select Edit, Properties. • Select the node, right-click and select Properties from the menu. • Double-click the node.	Set up Strategic Network Optimization nodes.

Setting up Strategic Network Optimization Nodes

Access the Strategic Network Optimization node window.

To set up a Strategic Network Optimization node:

1. Create a SNO node.
2. Connect the node to its input data object nodes and, optionally, its output data object nodes.
3. Select the SNO node, right-click, and select Properties from the menu.
4. In the SNO window, enter a name for the node in the Name field.
5. Select Input Data Objects. The attached input data object nodes are displayed in the Data Object column.
6. For each data object node, enter the path and name of the .smart file in the SMART Output column.
7. For each data object node, enter the path and name of the .view file in the View column.
8. Select Output Data Object. The attached output data object nodes are displayed in the Data Object column.

9. For each data object node, enter the path and name of the .smt file (smart graph) in the SMART Graph column.
10. For each data object node, enter the path and name of the .smart file in the SMART Output column.
11. Click Commands to display an edit window in which you can enter SNO batch commands. These commands are run after the model is loaded and the tables are imported, but before the model is solved. You can use this process to run the SNO import batch command that imports data into the model.
12. In the Model File field, enter the path and name of the SNO model that you want to use.
13. If you want the SNO interface to be displayed when you run the data flow, click Use GUI. Leave this option unselected if you do not want the interface to be displayed.
14. If you want the SNO model solved, click Solve Model. If you select Solve Model and Use GUI, the model is solved before the SNO interface is displayed.
15. Click Use GUI if you want to display the GUI.
16. Click Solve Model if you want to solve the SNO model.
17. Click Import/Export Only if you do not want to use a SNO license when the SNO node is run. If you do not use a license, you cannot save or solve a model, but you can use other functions, such as applying smart graphs. If you set Import/Export Only on, you cannot set Solve Model on.
18. Enter the input and output files.
19. (Optional.) Enter Tcl scripts for the node.

Creating and Managing Data Editor Nodes

This chapter provides an overview of Data Editor nodes, lists the window used to set up Data Editor nodes, and discusses how to set up Data Editor nodes.

Understanding Data Editor Nodes

This section provides overviews of:

- Data editor nodes.
- Displaying and printing data when you run data flows.
- How selection criteria in data object nodes affect the Data Editor.

Data Editor Nodes

You can use a Data Editor node to display the Data Editor; browse, edit, perform calculations on data, and produce Data Editor reports when you run a data flow.

A Data Editor node uses one input data object node and can have one output data object node. When a data flow is run and the data enters a Data Editor node, the Data Editor is started. Depending on the options that you select, the data is displayed, manipulated, or reported. The flow of data can be stopped while the Data Editor is running, or the flow can continue.

Displaying and Printing Data When you Run Data Flows

If you want the Data Editor to appear when you run the data flow, click Use GUI.

A report does not print automatically if both the Use GUI option and the Print Report option are selected in the Data Editor properties window. If you want to print a report automatically when you run a data flow, click Print Report but do not click Use GUI. If the Use GUI is selected, then from the File menu, select to print a report.

All active Data Editors shut down when you exit. To automatically print the information calculated by the Data Editor when you run a flow, you must click Hold Flow if the Data Editor does not have an output node. This process ensures that the Data Editor has sufficient time to print.

If you want the data displayed by the Data Editor to be read-only, click Read Only. If you do not click Read Only, you can change the data and save your changes.

Note

If the data object does not have a primary key, then you cannot change the data or save it, even if the Read Only button is not set.

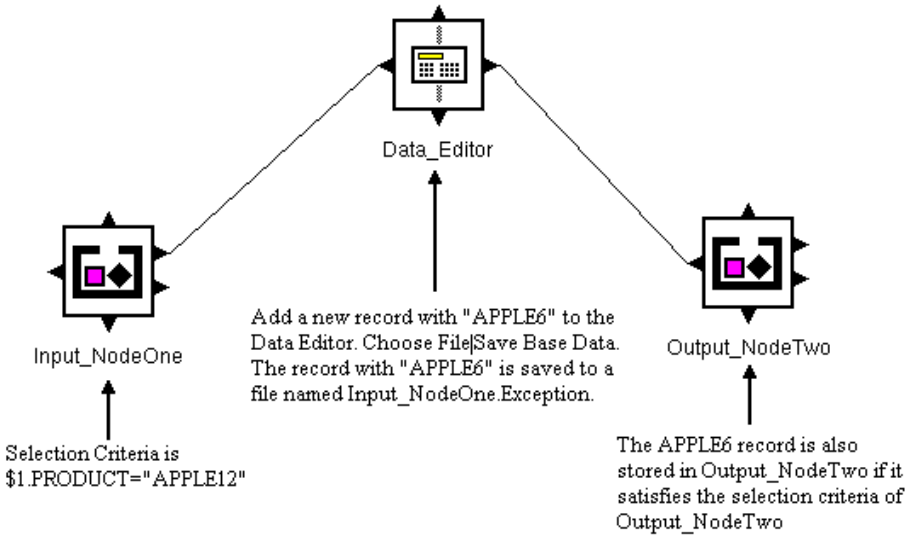
If you want the flow of data to stop when the Data Editor is displayed, click Hold Flow. The data flow stops until you exit from the Data Editor.

To use this option, the Data Editor node cannot have a node connected to its output attach point.

How Selection Criteria in Data Object Nodes Affect the Data Editor

If a Data Object node uses selection criteria and you select Browse Data from the View menu, the Data Editor only displays the data specified by the selection criteria. If you want to display different data, you must change the selection criteria expressions in the Data Object node window and re-open the Data Editor.

If you add a record in the Data Editor that does not satisfy the selection criteria in the Data Object node and select Save Base Data from the File menu, the behavior is different. Advanced Planning Agent tries to save this exception data to a new file in either the directory from which the Manager module is started, your home directory, or the tmp directory. If the save is unsuccessful, an error message appears. If the save is successful, Advanced Planning Agent names the file *xxx*.ExceptionData where *xxx* is the name of the Data Object node. The exception data is not saved to the data table as shown in the example below:



Example: Data Editor node

Windows Used to Set up Data Editor Nodes

Window Name	Navigation	Usage
Data editor node	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Do one of the following: • Select the node and select Edit, Properties. • Select the node, right-click and select Properties from the menu. • Double-click the node.	Set up Data editor nodes.

Setting up Data Editor Nodes

Access the Data Editor node window

To set up a Data Editor node:

1. Create a Data Editor node.
2. Connect the node to its input data object node and, optionally, its output data object node.
3. Select the Data Editor node, right-click, and select Properties from the menu.
4. In the Data Editor window, enter a name for the node in the Name field.
5. Ensure that the names of the input and output data object nodes, which are filled in automatically, are correct.
6. In the View File field, enter the name of the file that you want to use.
7. Select the Data Editor options that you want to use: Use GUI, Read Only, Hold Flow, Auto Save Base Flow, or Print Report.
8. If you click Auto Save Base Data and make changes to the data in the Data Editor, the data is automatically saved to the input data object when you exit from the Data Editor. The Save Base Data confirmation window is not displayed. To use this feature, you must click Use GUI. Do not select Read Only.
9. If the Auto Save Base Data is selected and your data table cannot be saved for some reason after you have made a change, the error is reported, but the Data Editor closes. You need to open the Error Browser to view the error message.
10. (Optional.) in the Tcl section, enter the Tcl scripts that you want to run before and after you import the data.

Setting up Shell Nodes

This chapter provides an overview of shell nodes, lists the window used to set up shell nodes, and discusses how to set up shell nodes.

Understanding Shell Nodes

This section provides overviews of:

- Shell nodes.
- Using parameters in shell nodes.

Shell Nodes

You can use a shell node in a data flow to run shell scripts or Tcl scripts.

Data does not flow into or out of a shell node. The position of a shell node, either above or below another node, determines whether its script is run before or after the node to which it is connected.

A Shell node starts a shell program, such as awk, sed, or Tcl, and the shell runs the script that is contained by the shell node.

For example, you may want to run a script that ensures that the dates after a particular join filter are no later than one day before the join occurred. The script would need to be run after the join filter, and you would position the shell node below the join filter.

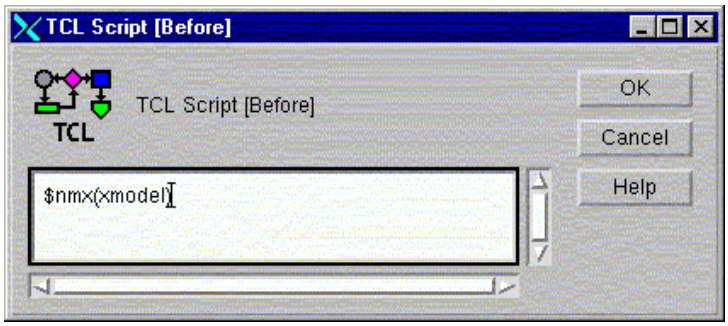
Using Parameters in Shell Nodes

You can now access your parameters from the shell node. This access can be useful to perform a backup of flat files before overwriting or to run a third-party system. For Windows NT, enter cmd /c in the Shell field or leave it blank.

Note

Parameters of this sort must have the NMX_ prefix as displayed in the following Shell Script window:

You use the parameters in a Tcl script as displayed in the following screen shot:



Tcl script

Tcl interprets a back slash as the beginning of an escape sequence. If you use back slash in Tcl, use double back slash instead. In directory paths on Windows NT, replace the back slash with either the forward slash or the double back slash.

Window Used to Set up Shell Nodes

Window Name	Navigation	Usage
Shell node	Select Manager, Data Flow in the APAg Manager project window. Select a data flow and click Edit. Do one of the following: • Select the node and select Edit, Properties. • Select the node, right-click and select Properties from the menu. Double-click the node.	Setting up Shell nodes.

Setting up Shell Nodes

1. Create a shell node.
2. Connect the node to another node.
3. Select the shell node, right-click, and select Properties for the menu.
4. In the Shell window, enter a name for the node in the Name field.
5. Enter a script in the Script area.
6. Optionally, in the Tcl Scripts section, enter the Tcl scripts that you want to run.
7. To specify the location of the shell program, In the Shell field, enter the path and name of the executable for the kind of script that you are using. For example, you might enter /bin/ksh or /bin/csh in the Shell field.
8. To enter a script, In the Script field, enter the script that you want to run.

Note

Back slashes at the end of lines are interpreted as a continuation of the command to the next line.

Running Data Flows

This chapter provides an overview of running data flows, lists the windows used to run data flows, and discusses how to:

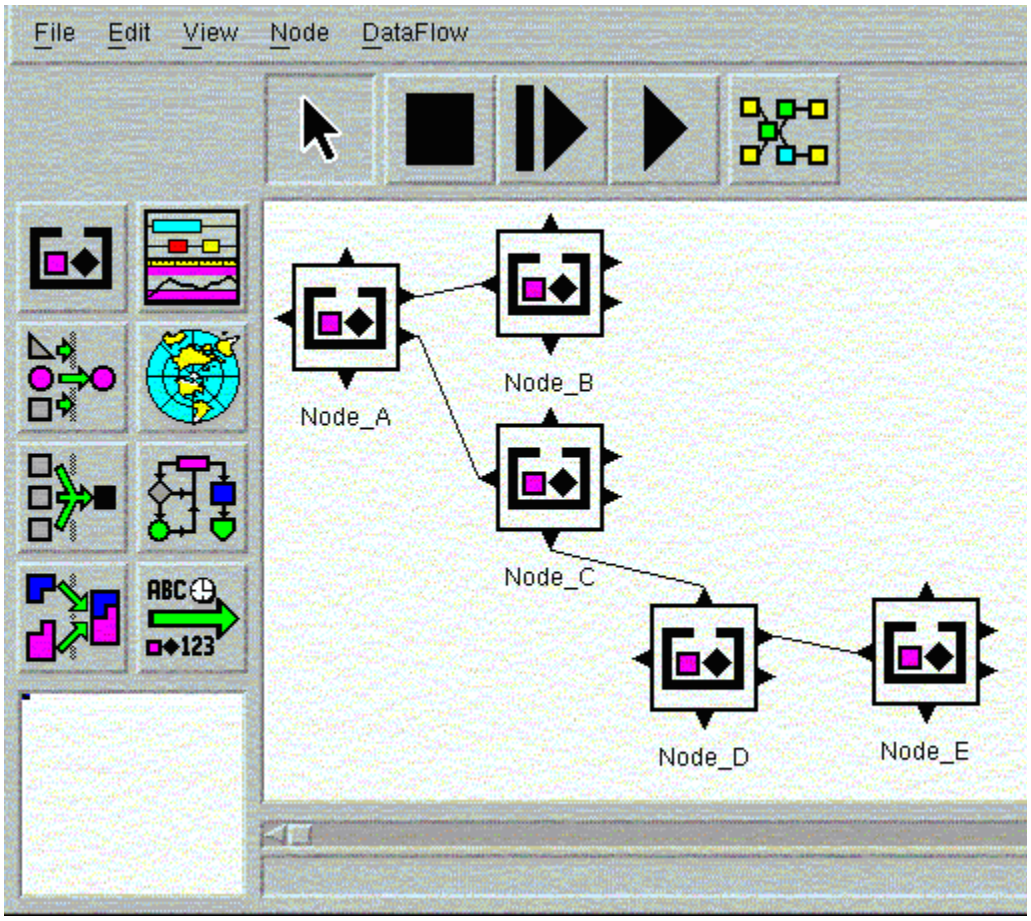
- Run data flows from the project window.
- Run data flows form the data flow launcher.
- Run data flows in batch.
- View and manage error messages.
- Organize log output.
- View and report data when you run data flows.
- Clear run information.
- Use arcs to map data types.
- Use parameters with data flows.
- Monitor systems for space.

Understanding Running Data Flows

When you run a data flow, Advanced Planning Agent manipulates data as it passes through the nodes that make up the data flow. The data is moved, manipulated, and optionally displayed as each node runs.

The overall flow of data is from left to right. When you run a data flow, the node on the left end of an arc runs before the node on the right. If nodes are attached vertically, the flow moves from top to bottom.

The following graphic shows a sample data flow:



Running data flows

In this example:

- Node A runs first because it is on the left end of the arcs that connect it to Node B and Node C.
- Node B and Node C are not connected so they can run in any order.
- Node C and Node D are connected and Node D is dependent on Node C. Node C runs before Node D.
- Node D runs before Node E because it is on the left end of the arc that connects them.

While a node is running, its name is boxed in yellow. After the node runs, its name is boxed in green.

If a data flow is larger than the data flow editor workspace, the view scrolls when you run the data flow so that you can see the node that is running.

If you run a data flow and errors are detected, the name of each node that contains an error is boxed in red. The red box is called an indicator. You can view the errors for each node that has an indicator.

When you run a data flow, data tables represented by data object nodes are saved only if the data flow completes successfully. If a data flow does not complete successfully, changes to the data are

discarded and the data remains as it was before the data flow was run. This safety feature prevents the overwriting of data with possibly incorrect data.

Note

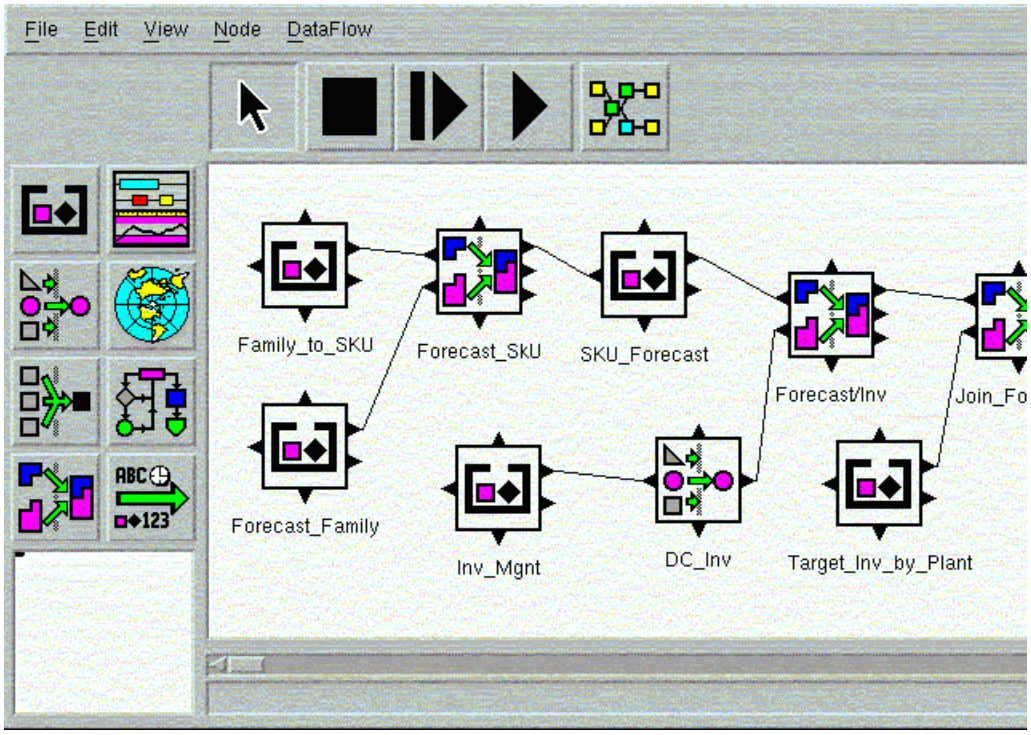
If a Strategic Network Optimization node, Production Scheduling - Process node, Data Editor node, or Shell node uses the data in a data object node, that data is always saved when the data flow runs.

If you want to make changes to a flow and run the selected part again, deselect the Run Selected Nodes icon or the Toggle Select Node, and select the part of the flow that you want to run. Otherwise, the changes that you make will not be refreshed in memory.

In Windows NT, file-locking can occur if you edit a data object definition while running a data flow.

You can run part of a data flow by selecting the nodes that you want to run. When you run that part of a data flow, the flow determines if there are nodes that supply data to any of the selected nodes. If so, those additional nodes automatically run.

Data is saved immediately as each node runs rather than waiting for the successful completion of the data flow. For example, in the following data flow, the join filter Forecast/Inv is selected as the node to be run:



Data flow with join filter selected to run

In this example, the Forecast/Inv filter requires data from the SKU_Forecast data object node and the DC_Inv selection filter to be run, and the DC_Inv selection filter requires data from the Inv_Mgmt data object node to be run.

Therefore, if the Forecast/Inv node is the selected node to be run, the Inv_Mgmt data object node, DC_Inv selection filter, and SKU_Forecast nodes are also run.

The nodes that are run in addition to the selected nodes go only as far back in the data flow as a previous data object node. The data in the data object node is assumed to be correct.

You can run any part of a data flow but depending on how you do so, the data in some nodes may become inconsistent with data in subsequent nodes. The flows keep track of these inconsistencies and may rerun nodes that were run previously.

For example, in the same data flow, if you wanted to run the data object node SKU_Forecast, the previous selection filter and data object nodes are also run. Once this process has occurred, it is possible that the data in the SKU_Forecast node is different from what it was when the subsequent nodes, such as the Forecast/INV join filter, were run.

You can run data flows in three ways:

- From the Advanced Planning Agent Manager.
- From the Data Flow Launcher.

  The Data Flow Launcher is similar in appearance to the Advanced Planning Agent Manager but does not need the licenses that are required to run flows in the Advanced Planning Agent Manager. Some functions in the Advanced Planning Agent Manager are not available in the Data Flow Launcher. For example, in the Data Flow Launcher, you can run data flows and view the results, but you cannot save any changes that you make to the data flows.

  You can use the Data Flow Launcher to browse data flows, browse data in data object nodes, and run data flows.

- In batch mode.

  Running data flows in batch involves running data flows in UNIX windows outside of the Advanced Planning Agent Manager. When you run a data flow in batch, the Advanced Planning Agent Manager is not displayed. You must create a batch file before you can run a data flow in batch. The batch file contains the instructions that specify the data flow to be run.

  For example, consider the following batch file:

  ```
  #Example of a batch file
  sci_LogMessage "Starting the batch data flow"
  sci_Login pat "abc123"
  sci_LoadProject MyPath/MyProject
  sci_DataFlow MyFlow run parameters "model /MyPath/MyFile"
  # The flow uses the dataflow called MyFlow.
  sci_LogMessage "The batch data flow is complete"
  ```

  When this batch file runs, the following actions occur:

  - The message confirming that the batch data flow is starting appears in the progress log file.
  - The user pat is logged in with the password abc123.
  - The project MyProject in the path MyPath is opened.
  - The data flow MyFlow runs. The data flow uses the parameter model with a value of /MyPath/MyFile, which identifies the location of the model.

- When the data flow is finished running, the following message appears in the progress log file: The batch data flow is complete.

Windows Used to Run Data Flows

Window Name	Navigation	Usage
Data Flow Manager	Select Manager, Data Flow. Select the data flow that you want to run and click Run.	Running data flows.
Project	Select Manager, Data Flow. Select the data flow that you want to run and click Edit.	Running data flows from the project window.

Running Data Flows from the Project Window

Access the Project window.

To run a data flow from the project window:

1. (Optional) To run only part of the data flow, select the nodes that you want to run.
2. Do one of the following:
   - To run the data flow, select Data Flow, Run.
   - To run the data flow one node at a time, select Data Flow, Step.

     Repeat for each node.

     To stop running the data flow one node at a time, select DataFlow, Terminate Step Mode.

Running Data Flows From the Data Flow Launcher

To run a data flow from the Data Flow Launcher:

1. Do one of the following:
   - On Windows NT, login with your userid.

     Select Start, then select Programs, then select Supply Chain Planning, then select Advanced Planning Agent *x.x.x* , and then select Data Flow Launcher.

   - In a UNIX window, enter apag.

     Type 3 and press Enter.

     Select the version that you want to use.
2. In the Name field, enter your userid.
3. In the Password field, enter your password.

4. Click OK.
5. Select File, Open.
6. Select a project and click OK.
7. Select Manager, Data Flow.
8. Select a data flow and click Run.

Running Data Flows in Batch

This section discusses how to:

- Create batch files for running data flows.
- Run batch files.
- Run batch files with parameters.

Creating Batch Files for Running Data Flows

To create a batch file:

1. Using a text editor such as vi or emacs, create a new file.
2. Enter the following command:

   ```
   sci_Login username "password"
   ```

   Where:

   - *username* is a valid userid.
   - *password* is the password for the userid. The password must be enclosed in quotation marks.
3. Enter the following command:

   ```
   sci_LoadProject path / project
   ```

   Where *project* is the name of the project that contains the data flow that you want to run and *path* is the complete file path.

4. Enter the following:

   ```
   sci_DataFlow dataflow run parameters "variables"
   ```

   Where *dataflow* is the name of the data flow that you want to run and *variables* are the variables for the parameters used. The parameters must be enclosed in quotation marks. Enter the *parameters* keyword only if you want to specify any *variables.*

5. (Optional.) Enter the following:

   ```
   sci_LogMessage "message"
   ```

   When you run the batch file, the *message* is recorded in the *sci_username.log* file. The message must be enclosed in quotation marks.

You can include comments in the batch file. Comment lines must begin with the # character.

6. Save the batch file and exit from the text editor.

Running Batch Files

Open a UNIX window.

Note

The Advanced Planning Agent Manager does not need to be running before you run a batch file.

To run a batch file, enter the following command:

```
SCI_FRED_x.x.x -batch batchfile
```

Where *x.x.x i*s the version number that you want to run, and *batchfile* is the name of the batch file.

If you are using a runtime license, use the following command instead: `SCI_RUNTIME_x.x.x`

Running Data Flows in Batch Using Parameters

The syntax for running a data flow in batch using parameters is as follows:

```
sci_DataFlow dataflow run parameters "name value"
```

Where:

- *dataflow* is the name of the data flow.
- *name* is the name of the parameter.
- *value* is the value of the parameter.

If you have more than one parameter, use the following syntax:

```
sci_DataFlow dataflow run parameters { name1 value1 name2 value2... }
```

Where:

- *dataflow* is the name of the data flow.
- *name1* is the name of the first parameter.
- *value1* is the value of the first parameter.
- *name2* is the name of the second parameter.
- *value2* is the value of the second parameter.
- ... represents further pairs of parameter names and values, each of which must be enclosed in a set of brace brackets.

Specifying the Repository and Resource Area When Running Data Flows in Batch

You can specify the repository and resource area that you want to use when you run a batch file, as follows:

SCI_FRED_*x.x.x* -NMX:NMX_INIT= *repository* -NMX:NMX_RES= *path* -batch *batchfile*

If you are using a runtime license, use *SCI_RUNTIME _ x.x.x,* instead of the above command.

Specifying the Advanced Planning Agent Version Number Before Running Batch Files

You can simplify the command to run a batch file by specifying beforehand what the version number is. After specifying the version number, you do not need to include it as part of the batch command. You specify the version number by creating an alias.

To create an alias for the version number, include the following command in your .profile file:

```
alias sci_BATCH='SCI_FRED_ x.x.x -batch'
```

Where *x.x.x* is the version that you want to run.

To set the version number for a single UNIX window for as long as the window is open, enter the command in a UNIX window rather than putting it in your .profile file.

If you are using a runtime license, use the SCI_RUNTIME_ *x.x.x* command instead.

Running Batch Files after Specifying the Advanced Planning Agent Version

To run a batch file after specifying the Advanced Planning Agent version, enter the following command in a UNIX window:

sci_BATCH *batchfile*

Where *batchfile* is the name of the batch file

Passing Environment Variables to Scripts

The following sample script shows how you can set environment variables in a batch file script and pass the variables to another script that uses them as parameters.

```
#!/bin/ksh
# The NMX_RES and NMX_INIT can alternatively be set
# in the users' profile.
export NMX_INIT=production
export NMX_RES=/usr/APAgdir/vers_ x.x.x /cfg
# Export the directory in which the batch files reside and the
# one that holds the projects.
```

```
export BATCHDIR=/user/banana/sci/sci_batch
export PROJECTS=/user/banana/sci/scisci_project
# this script uses the environment variables as parameters
#!/bin/ksh
# The NMX_RES and NMX_INIT can alternatively be set
# in the users' profile.
export NMX_INIT=production
export NMX_RES=/usr/APAgdir/vers_ x.x.x /cfg
# Export the directory in which the batch files reside and the
# one that holds the projects.
export BATCHDIR=/user/banana/sci/scibatch
export PROJECTS=/user/banana/sci/sci_project
# Export the values for the two environment variables. These
# variables will be used to pass the values to the SCI batch
# file that runs the flows.
export RUNDATE='date +%Y/%m/%d'
export MODEL_CODE="10NJA"
# Run the SCI batch file using the runtime license. Change RUNTIME
# to FRED if you have a development license.
/usr/bin/SCI_RUNTIME_3.1 -batch $BATCHDIR/production.batch
# The batch file uses the environment variables as parameters
# Grab the date, the model code and the project's location values
# from the appropriate environment variables that have been
# set in the macro script and place them in the associated Tcl
# variables. If you are not using the values of environment
# variables, you can hardcode the appropriate values when running
# the flow with parameters further down.
set R_Date $env(RUNDATE)
set Model_Code $env(MODEL_CODE)
# Get the project location that was defined in the macro file.
set ProjDir $env(PROJECTS)
# Log diagnostic messages in the progress log.
sci_LogMessage "--------------> Starting production flow"
sci_LogMessage "Date value is: $R_Date"
# Log in as user1
sci_Login user1 "password"
# Load the project
sci_LoadProject $ProjDir/production.prj
# Run the Production flow using the values from R_Date and
# Model_Code.
# The flow has two flow parameters defined: Run_Date and MODELCODE.
set flow {sci_DataFlow "Production" run parameters "
Run_Date $R_Date
MODELCODE $Model_Code
"
}
eval $flow
# Log a message for the completion of the flow.
sci_LogMessage "Production flow completed."
```

Verifying that Data Flows Have Run in Batch

The following sample batch file shows how you can verify that a flow in batch mode has run successfully:

```
sci_LogMessage "Beginning Batch Run"
sci_Login username "password"
```

```
sci_LogMessage "Completed Login"
sci_LoadProject /path/project
sci_LogMessage "Loaded Project File" sci_LogMessage
set y [sci_DataFlow dataflowname run parameters "variables"]
puts "the return code = $y"
sci_LogMessage "Completed Batch Run "
```

The 6th and 7th lines of code catch the return code. If the flow runs without generating an error, the variable "y" returns " 0 {}". If the flow generates an error, the variable "y" returns "1 {There was a problem executing the current step}".

If a crash occurs, the control does not return to the Manager, and no value is passed to "y". You can check for these situations by using a statement such as sci_LogMessage "Completed Batch Run" in your batch files after each dataflow runs. This message is directed to the sci_ username logfile. You can then check that file to see how many messages were printed which determines where the crash occurred.

Viewing and Managing Error Messages

This section provides an overview of error messages and discusses how to:

- View error messages after running data flows.
- Clear error indicators.
- View error messages.
- Specify errors that you want to view.
- Compress identical errors into one message.
- Highlight errors.
- View errors individually.
- Refresh error data.

Understanding Error Messages

If you run a data flow and errors are detected, the name of each node that contains an error is boxed in red. The box is called an indicator. You can view the errors for each node that has an indicator.

Errors are tracked in your error log. You may find it useful to display the contents of your error log for troubleshooting or reporting problems. You can view your error messages in the Log Browser window, and you can manipulate the format in which your errors are displayed.

You can highlight error messages in red based on the severity level of the message. When you select a severity level to be highlighted, all messages at that level and above that level are highlighted.

Viewing Error Messages After Running Data Flows

To view error messages for a node:

1. Select a node.

2. Select View, Browse Error.

The Error Report window displays the error messages for the selected node. While the Error Report window is displayed, you can select another node to display its error messages.

Clearing Error Indicators

After you run a data flow, the nodes with errors are boxed with indicators.

To clear error indicators, do one of the following:

- To clear the error indicator from one node, select the node and then select View, Clear Indicator.
- To clear all error indicators from a data flow, select View, Clear All Indicators.
- To clear all error indicators and reset the run selected mode, select DataFlow, Clear Run Info.

Viewing Error Messages

To view error messages in the Log Browser, In the data flow editor, select Browse Logs from the View menu.

To find and view your error log manually:

1. Change to the resource area directory by entering the following:

   ```
   cd $NMX_RES
   ```

   If the NMX_RES is not set, change directories to the absolute path of the directory to which you usually export the NMX_RES variable.

2. Display the contents of the *errmsg.cfg* file by entering the following:

   ```
   more errmsg.cfg
   ```

3. In the file is an entry similar to the following, where *path* is a directory path:

   ```
   < nmxerrlog > / path / logs
   ```

4. Change to that directory by entering the following:

   ```
   cd / path / logs
   ```

5. List the contents of the directory by entering the following:

   ```
   ls
   ```

   Each user has an error log file called *name .log*, where *name* is the user's login name.

6. Display the contents of your error log by entering the following:

   ```
   more name .log
   ```

   The entries in the error log have the following formats and meanings:

   - *Date and time* specifies when the error occurred.
   - *Error message* is the explanation of the error.
   - Message: < *message_number* > is useful when reporting problems to support staff.
   - Severity: < *severity_level* > specifies severity level of the error.

- File: < *file_affected* > is useful when reporting problems to support staff.
- Line: < *line_in_file* > is useful when reporting problems to support staff.

Note

If a data table contains bad data, the first erroneous record is logged in the error log. For example, if two records in a data table have the same primary key, the second record (the record that violates the primary key restriction) is logged when the flow attempts to load the data table.

Specifying Errors that You Want to View

To specify the errors that you want to view:

1. Select Display, Filter.
2. Enter a word or text string in the Message field.

   The Error Browser does a text search and filters in only the messages with this text. For example, if you only want to view messages associated with expressions, enter expression in this field. The Error Browser only displays messages that contain this word.

   Leave this field blank if you do not want to restrict your search by "Message".

3. Click Severity and select a severity level from the option menu (All, Information, Warning, Error, Abort, or Unknown).
4. Click the Time Stamp field and enter any time information that pertains to the information that you want to view.

   For example, if you only wanted to view errors that occurred on February 5, 2005 enter 20050205. Leave this field blank if time information is not necessary. Usually, no reason exists to use the File Name or the Line Number fields except for debugging information when making a support call.

Compressing Identical Errors Into One Message

To compress identical errors into one message, select Display, Compress.

To view your log file:

1. Select Display, Log File.
2. In the Log File window, click the top button and select one of the following from the menu:
   - Error Log, to display your error log file
   - Progress Log, to view progress diagnostics
   - User Log, to view your user log file

Highlighting Errors

To highlight errors, select Display, Highlight, and one of the following:

- None
- Warning
- Error
- Abort

The default is to highlight the Error level messages.

Viewing Errors Individually

To view the errors individually:

1. Do one of the following:
   - Select an error message and select Display, Show Detail.
   - Double-click an error.
2. In the Error Detail window, click the More>> option to show the time stamp, file name, and line information.
3. Click Close to return to the Log Browser window.

Refreshing Error Data

The Refresh From Log option reads and displays error messages from the Log File Browser. You can use the refresh option to view any errors that have occurred since you opened the Error Browser window.

To refresh your data, from the Display menu, select Refresh From Log.

Organizing Log Output

This section provides an overview of log output, lists the window used to manage logs, and discusses how to:

- Change the location for log output.
- Change the maximum log size.
- Change the truncate value of logs.

Understanding Managing Logs

A LogManager window exists at the project level and the data flow editor level. In the LogManager window you can do the following:

- Specify the location of error and progress log output.
- Specify the maximum size of the log in bytes.
- Specify whether the log can be truncated and the percentage to which it may be truncated.

Note

Changes made in the project level LogManager window become the default settings in the data flow editor window. You must change the settings in the data flow editor LogManager window to override the project level LogManager window.

Window Used to Manage Logs

Window Name	Navigation	Usage
LogManager	In the Project window, select File, Logs.	Managing log files.
LogManager	In the data flow editor, select DataFlow, Logs.	Managing log files.

Changing the Location for Log Output

To change the location for log output:

1. Access the LogManager window.
2. Click Location.
3. In the blank Location field, enter the directory path to your new log output location.
4. Click OK.

Changing the Maximum Log Size

To change the maximum size of logs:

1. Open the LogManager window.
2. Click Max Size.
3. In the blank Max Size field, enter the new maximum size (in bytes) for the logs.
4. Click OK.

Changing the Truncate Value of Logs

To change the truncate value of logs:

1. Open the LogManager window.
2. Click Truncate To.
3. Click the Never Truncate button.
4. From the menu, select a value.

   For example, if you select 90%, then 10% of the log file is truncated. If you select 0%, the log file is deleted when the maximum size is reached.

5. Click OK.

Note

The Never Truncate option overrides the Max Size option.

Viewing and Reporting Data When You Run Data Flows

This section discusses how to:

- Display the Data Editor, Production Scheduling - Process, or Strategic Network Optimization.
- Refresh the data flow editor.
- Print reports from the Data Editor.

Displaying the Data Editor, Production Scheduling - Process, or Strategic Network Optimization

When you run a data flow, the GUI of the Data Editor, the Production Scheduling - Process system, or the Strategic Network Optimization system can be displayed. If you display one of these systems, the data flow stops until you exit from the Data Editor, Production Scheduling - Process, or Strategic Network Optimization. When you exit, the data flow resumes with the next node in the data flow.

When the GUI is displayed, you can use the product as if you had started it on its own. For example, you can run an algorithm in the Production Scheduling - Process system before exiting.

To display the Data Editor, Production Scheduling - Process, or Strategic Network Optimization, in the applicable node's properties window, click Use GUI.

Refreshing the Data Flow Editor

To refresh the data flow editor, select Refresh from the View menu.

Printing Reports from the Data Editor

When you run a data flow, you can automatically print a report from the Data Editor.

To print reports from the Data Editor, in the Data Editor properties window, click Print Report.

The report that is printed depends on the view file that you selected in the Data Editor node.

Clearing Run Information

To clear the run information, select DataFlow, Clear Run Info.

Using Arcs to Map Data Types

You can use an Arc Properties window to write data to a Data Object node. In the Arc Properties window, you can define how data is copied from the input node to the output node. The mode selected in the Arc Properties window overrides the option selected in the data object properties window.

To display the Arc Properties window

1. Select an arc.
2. Do one of the following:
   - Select Edit, Properties.
   - Right-click, and select Properties from the menu.
3. In the Arc Properties window, select one of the following four options:
   - The No Action option defaults to the mode selected in the data node properties window.
   - The Copy by column position option copies by position. In this case, the input data records and the output data records must contain fields of the equivalent type in the same position. Data in corresponding positions are copied from input to output records. Field names are ignored, but definitions must match exactly.
   - The Copy by column name option copies by the field names. Fields contained in the input data records and not in the output data records are ignored. Fields contained in the output data records, but not in the input data records, are given the value of null.
   - The Copy by special map option allows you to define how the data is copied. To open the Copy Map Definition window, click Copy by special map and then click Define Map.

   The field names from the input data table are displayed in the From Field List column. The field names from the output data table are displayed in the Copy To column. Click both mouse buttons and drag the data types from the From Field List to the Copy From fields.

Note

Ensure that the data types in the Copy From fields are equivalent to corresponding Copy To fields.

Using Parameters with Data Flows

Parameters are variables that represent explicit data, such as file paths or field names. You can specify parameters at the project level or the flow level. Flow level parameters override project level parameters.

This section discusses how to:

- Add parameters at the project level.
- Add parameters at the data flow level.
- Edit parameter names.

- Edit parameter default values.
- Delete parameters or default values.
- Override default parameters when you run data flows.

Adding Parameters at the Project Level

Access the Project window.

To add parameters at the project level:

1. Select File, Parameters.
2. In the Project Parameters window, enter the name of the parameter in the Parameter column. Optionally, enter the default value of the parameter in the Default Value column.
3. Click OK.
4. To save the parameter for future use, select File, Save.

Adding Parameters at the Data Flow Level

Access the Data Flow editor.

To add parameters at the data flow level:

1. Select Parameters, DataFlow.
2. In the Data Flow Parameters window, enter the name for the parameter in the Parameter name column. Optionally, enter the default value of the parameter in the Default Value column.
3. Click OK.
4. Select File, Save to save the parameter for future use.

Editing Parameter Names

Access the Data Flow Parameters or Project Parameters window.

To edit a parameter name:

1. Highlight the parameter name that you want to edit.
2. Enter your change in the Parameter column.
3. Click OK.
4. (Optional.) If you want the changes to apply for future use, save the project and the flow before you exit.

Editing Parameter Default Values

Access the Data Flow Parameters or Project Parameters window.

To edit a parameter default value:

1. Highlight the default value and press either the keyboard Backspace or Delete key.
2. Enter the new value or, optionally, leave the cell blank. If you leave the cell blank, the value is set to null.

Deleting Parameters or Default Values

Access the Parameters window.

To delete a parameter or a default value, do one of the following:

- Click Delete to delete a specific parameter and default value.
- Click Delete All button to delete all entries from both columns.

The system prompts you for confirmation before the deletion occurs. Click Cancel to close the Parameters window and cancel any current changes that you do not want to save.

Overriding Default Parameters when you Run Data Flows

Access the Data Flow Editor window.

To override default parameters when you run a data flow:

1. Select DataFlow, Parameter Prompt.

   When you start to run the flow, the Run Time Parameters window appears before the flow actually runs.

2. You can override the Default Value by entering a value in the Current Value field. You can add, edit, or delete a current value.

   The Current Value parameters can be used for more runs until you exit from the data flow. They are deleted once you exit and cannot be saved with the flow.

3. If you want to apply the same Current Value parameters to more runs, you can turn the Parameter Prompt option off so that the Run Time Parameters window is not displayed each time that you run a flow. The Current Value parameters remain applicable until you change them or exit from the flow.
4. If you want to change the Current Value parameters, turn the Parameter Prompt option on, and the Run Time Parameters window appears each time that you run a flow.

Note

To display the Run Time Parameters window, the flow must have default parameters.

Specifying Parameters In a Properties Window

You can enter a parameter created in any field within a node properties window. Use the syntax *${ parameter }* where *parameter* is the name of the parameter.

Specifying Parameters In a Selection Criteria Window

You can enter a parameter created in any field within a selection criteria window using the syntax *${ parameter }* where *parameter* is the name of the parameter.

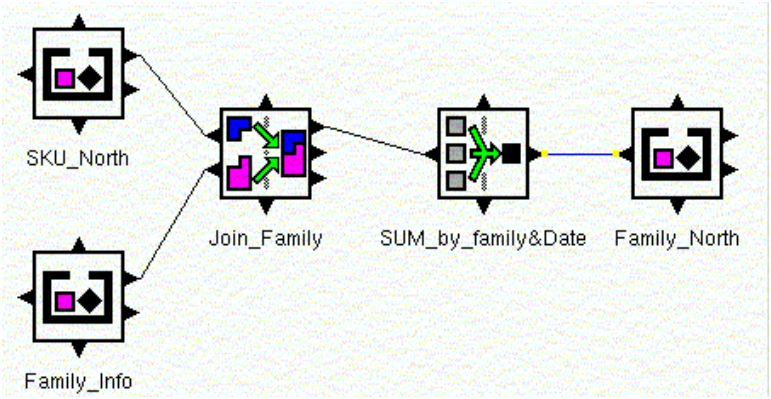
If your parameter is a text string, such as the name of a database table field, enclose the entire parameter in double quotation marks. For example, if you create a parameter called *family_name* that uses data from the field that contains product family names, enter:

"${family_name}"

If the parameter is a number, such as demand values, you do not need to use quotation marks. For example, if you create a parameter called *demand_values* that uses numerical data, enter:

${demand_values}

The data flow example below joins and aggregates family data and SKU demand in the north. It results in the data for demand by family.



Selection criteria node example

A parameter is used to find out the northern demand data for only one family at a time. The parameter *family_name* is entered as part of an expression in the output selection criteria in the aggregation node. The expression states that from the data field called family, the output should look only for the information specified by the parameter.

Monitoring Systems for Space

The amount of paging space that is required depends on the type of processes you are performing on your system. If available paging space falls below a threshold (known as the paging-space warning level), the system normally informs all the processes of this condition. If the shortage continues and falls below a second threshold (known as paging-space kill level), the operating system sends the SIGKILL signal to the processes that are the major users of the paging space until the number of free paging-space blocks is above the paging-space kill level. Advanced Planning Agent is generally a major user of paging space. If a lack of swap space is suspected for Advanced Planning Agent, re-run your data flows and monitor the swap space using the following platform specific commands or methods:

Platform	Command or Method
HP	`Swapinfo –a`
AIX	`Isps –a`
Windows	Use Task Manager's Performance Monitor.

To prevent Advanced Planning Agent from crashing, advise your system administrator if the available swap space is 10 percent or less.

Creating the Definition File in the PSI Editor

Access the PSI Editor.

To create the definition file in the PSI Editor:

1. Select File, Smart.
2. In the Smart Definition window, in the Object Type field, click and drag the left mouse button to select the type of definition file that you are creating.

   You can select from Resource, Inventory, or Operation. You can only import or export data to one object type per definition file.

3. In the Definition File field, enter the directory path and filename that you want to contain the definition file.
4. Select Save.

Viewing the Smart Definition File

You can run the SMARTOUT command from the Production Scheduling - Process command editor to view the output data. For the files created in this example, you enter the following command, which specifies the smart definition file *inventory.smart* and the output file *inventory.output* :

smartout inventory.smart inventory.output

Exported PSI information is written to a data file. The file contains a header that tells you the meaning of each piece of data that follows. The header looks similar to the following:

```
%%BeginProlog
%%FieldNames: <inv> <Start> <End> <Starting Inventory> <Total Demand>
<Total Supply/Demand> <Produced> <Ending Inventory> <Safety Stock>
<AVGDEM>
%%FieldTypes: string date date real real real real real real real
%%FieldKeys: key key key data data data data data data data
%%FileType: delimited
%%FieldDelimiter: <tab>
%%Creator: username
%%CreationDate: 2005/07/26/10:23:17
%%CreatedBy: Production Scheduling - Process
%%EndProlog
```

Note

Dates appear in the output file in 4-digit format, such as 2005.

You need this information to create a data object definition in Advanced Planning Agent.

Using the Production Scheduling - Process Node in Advanced Planning Agent

Assuming such a storage definition exists, you can exit from the Production Scheduling - Process system and start the Advanced Planning Agent Administrator to create the appropriate data object definition.

Take note of the type selected for each field and the format entered for the data fields. The field types and formats must be correct, or the data cannot be used correctly in the data flow.

Once you have created the data object definition, you can start the Advanced Planning Agent Manager and create the part of the data flow that integrates the Production Scheduling - Process system.

Select the Production Scheduling - Process node and select Properties from the Advanced Planning Agent menu. The Production Scheduling - Process window appears.

In the Case File field, enter the name of the Production Scheduling - Process case. In the Definition File column, enter the name of the smart definition file that you created in the PSI Editor.

You can now run the data flow. The Production Scheduling - Process system starts and automatically exports the data from the case to the output data object that you created. You can view the data in the Data Editor and manipulate it by adding more nodes to the data flow.

Specifying Input and Output Definition Files

To use the information in a data object node that is attached to it, the Production Scheduling - Process node requires a smart definition file. A smart definition file converts information into a format that the Production Scheduling - Process system can import or export.

You can create smart definition files with a text editor or with the PSI Editor of the Production Scheduling - Process system.

To specify an input or output definition file:

1. Do one of the f ollowing:
    - To specify an input definition file, select Input Data Objects.
    - To speicfy an output definition file, select Output Data Objects.
2. Enter the definition file for each data table in the Definition File column.

Using Tool Command Language

This chapter provides an overview of Tool Command Language (Tcl), and discusses how to:

- Create Tcl scripts in data flows.
- Troubleshoot data flows.

Understanding Tool Command Language

You can use Tool Command Language (Tcl) in Advanced Planning Agent. Tcl is a scripting language that you can use to control and extend Advanced Planning Agent features.

For most kinds of nodes, you can run a Tcl script just before or just after the node runs. You can also run a Tcl script in a shell node.

The following tables lists some uses of Tcl:

Use	Description
Flow control	If commands and control loops (such as while and for)
List manipulation	Create, search, and sort
Expressions	Using arithmetic operators (+ - /), relational operators (> < =) and logical operators (and, or, not) with whole or real numbers
String manipulation	Pattern matching, searching and comparing strings, and extracting substrings
I/O operations	Printing to and reading from the operating system or a file
Variables	Support for simple variables and arrays (multidimensional variables)

Note

If a Tcl script contains the percentage sign (%) and has a syntax error, instability might occur in the Advanced Planning Agent Manager.

Creating Tcl Scripts in Data Flows

The following nodes have a Tcl Scripts section in their properties window that you can use to run a Tcl script before and after the node runs:

- Filter nodes: selection, aggregation, and join.
- Production Scheduling - Process nodes.
- Strategic Network Optimization nodes.

- Data Object nodes.
- Data Editor nodes.
- Shell nodes.

To create a Tcl script in a data flow:

1. Select a node and click Properties.
2. In the Tcl Scripts section of the node's properties window, click Before or After.

   If you select Before, the Tcl script runs after the input data is received and before the node performs its function. If you select After, the Tcl script runs after the node performs its function and before the output data is sent to another node.

3. In the Tcl Script window, enter the script in the window and click OK.

In Tcl 7.5 and later versions, anything that starts with 0-for example, 08 or 09-is treated as an octal number. As a result, scripts that perform calculations or comparisons should be checked to make sure that they work as you intend them to work, particularly when doing mathematical calculations using dates or times.

You can replace the *expr* command with another Tcl procedure that avoids this problem. Create the following procedures and store them in your Tcl directory:

```
proc noLeadZ {num} {
  set newnum [string trimleft $num 0]
  if { $newnum ==""} {
  return 0
}

   return $newnum
   }
```

If you create a Tcl script in a Shell node or Tcl Script window and the brace brackets *{ }* are not correctly balanced, you cannot save the script. A warning message appears. If you want Advanced Planning Agent to automatically add brace brackets, click Yes. This procedure allows you to save the script, but does not correct syntax errors in the script. If you want to correct the script yourself, click No.

Troubleshooting Data Flows

You can use the following Tcl commands in a data flow for troubleshooting or debugging:

```
sci_dumpTupleSet [ nodename [ attach name [ filepath/name count]]]
```

The above command returns the rows of data at a particular point in a data flow. The *nodename* is optional. By default it is the node from which the Tcl script runs. If you want to write the output to a file, specify *output filepath/name* ; otherwise, the data is written by default to standard output. The *count* , which is required, specifies the number of records that you want to output. Specify *0* to return all the records:

```
sci_tupleCount [ nodename ]
```

The above command returns the number of records in a node. The *nodename* is optional. By default it is the node from which the Tcl script runs.

Note

Elements within brackets [] are optional.

Viewing and Editing Data

The Data Editor displays data in a spreadsheet format. You can use the Data Editor to:

- Examine data.
- Edit data.
- Perform calculations on data.
- Customize the display of your data.
- Produce and print reports.

You can view and edit data in the Data Editor. You can open the Data Editor in the following ways:

- From the Advanced Planning Agent Manager
- From a data object node in the data flow editor
- From the Data Editor node

This chapter discusses how to:

- Customize views.
- Work with data.
- Select data classifications and types.
- Use breaks.
- Use expressions.
- Assign values to fields.
- Save views.
- Save to database tables.
- Export data to files.
- Use the inverse formula to automatically update data.

Customizing Views

You can change the appearance of the Data Editor in several ways to display your data in a way meaningful to you. After you change the Data Editor's appearance, you can save your changes as a view.

This section discusses how to:

Move rows and columns.

Swap rows and columns.

Hide and reveal rows and columns.

Show data in layers.

Move data in layers.

Display headings.

See Also

Saving Views

Moving Rows and Columns

Access the Data Editor.

To move a row or column, drag the row or column label to the label of the row or column where you want to move the row or column.

Swapping Rows and Columns

Access the Data Editor.

To swap rows and columns, select Configure, Swap Data Row/Column.

Hiding and Revealing Row and Column Labels

Hiding row and column labels is useful if you have many different labels and you want to display only a few of them.

Access the Data Editor.

To hide or reveal row and column labels, do one of the following:

- To hide row or column labels, click the label that you want to hide.
- To reveal a hidden row or column label, click the hidden label.

Showing Data in Layers

You can show data in a series of layers using the Z dimension feature. The Z dimension allows you to move through the layers of the Data Editor in a way that is similar to turning the pages of a book. Each layer, or page, shows different information, and you can move forward and backward through the layers.

If you want to display the information by period and one period at a time, use the Z dimension feature. A separate layer or page exists for each period.

Access the Data Editor.

To show data in layers:

1. Position the cursor on the label that you want to use to layer your data.
2. Press and hold the left mouse button.

3. Position the hand icon on top of the Z dimension button and release the mouse button.

   The label that you moved is now displayed on the Z dimension button. You can move more than one field to the Z dimension.

4. To move through the available displays, do one of the following:
   - Click the up or down arrow.
   - Click a tab at the bottom of the window.

Moving Data from Layers

Access the Data Editor.

To move data from a layer:

- Position the cursor on the label in the Z dimension that you want to relocate.
- Press and hold the left mouse button.
- Position the pointing hand icon on a row or column label on the left side of the Data Editor, and release the mouse button.

Displaying Headings

Access the Data Editor.

To display headings for each field:

- Select Configure, Display.

  In the Configure Data window, do one or more of the following:
  - To display headings for each field, select Show Duplicate Labels.
  - To display the complete heading only once, unselect Show Duplicate Labels.
  - To display the system title, select the Show System Title option.
  - To display the user title, enter a title in the User Title field.

Working with Data

You can make the following changes to your data:

- Adding records.

  When you add a record, you can edit information in the new record. The data in the new record is identical to the record that is selected when you create a new record. You can create only one record at a time. When you create a record, it must be unique. The Data Editor does not allow you to create identical records.You cannot add a NULL value through the Data Editor.

- Adding rows.

When you add a row, you you can edit information in the new row. The data in the new row is identical to the data row that is selected when you add a row. You can add only one data row at a time. Each new data row must be unique. The Data Editor does not allow you to create identical data rows.

- Adding columns.

  When you add a column, you can edit the information in the new column. The data in the new column is identical to the column that is selected when you add a column. You can add only one column at a time. When you create a column, it must be unique. The Data Editor does not allow you to create identical columns.

- Deleting records.
- Deleting records when data tables have only primary keys.
- Deleting rows.
- Deleting columns.
- Changing the date format.
- Defining the number of decimal places.
- Changing values in rows, columns, or views.

  You can change all the values in a specific row or column or throughout the entire Data Editor. For example, if you want to examine the effects of increased demand on your inventory levels, you can uniformly increase the demand. You could then apply an expression to recalculate the inventory required to meet the increased demand.

Adding Records

Access the Data Editor.

To add a record:

- Select a cell.
- Select Edit, Add, Record.
- In the Add Record window, change some or all the record information so that the new record is unique.
- Click OK.

Adding Rows

Access the Data Editor.

To add a row:

- Select a cell in the row.
- Select Edit, Row, Add.
- In the Add Record window, change some of the row information so the new row is unique.

- Click OK.

Adding Columns

Access the Data Editor.

To add a column:

- Select a column or a cell in a column.
- Select Edit, Add, Column.
- In the Add Column window, change some or all of the column information so the new column is unique.
- Click OK.

The new column appears to the right of the selected column.

Deleting Records

Access the Data Editor.

To delete a record:

- Select the cell that you want to delete.
- Select Edit, Delete Record.
- In the Delete Record window, click OK.

Deleting Records When Data Tables Have Only Primary Keys

Access the Data Editor.

To delete a record when a data table has only primary keys:

- Select Configure, Fields.
- In the Configure Fields window, select a primary key, and from the menu, select dataReadWrite.
- Click Apply & Close.

Deleting Rows

Access the Data Editor.

To delete a row:

- Select a cell in the row that you want to delete.
- Select Edit, Delete, Row.
- In the Delete Row window, click OK.

Deleting Columns

Access the Data Editor.

To delete a column:

- Select a cell in the column that you want to delete.
- Select Edit, Delete, Column.
- In the Delete Column window, click OK.

Changing the Date Format

Access the Data Editor.

To change the date format:

- Select Configure, Display.
- Enter the date format that you want in the Date Display Format field.

  If the Data Editor does not understand the format, it displays the default format (day, month, year). The following table lists the available date formats:

Date Format	Description
%d	Day as a number-for example, 19.
%b	Month as a three-letter abbreviation-for example, Nov.
%m	Month as a number, for example-11 for November.
%y	Year as a two-number abbreviation-for example, 00.
%B	Month by its full name, for example-November.
%D	Entire date in the mm/dd/yy format-for example, 11/19/05.
%Y	Year in full-for example, 2005.

You can combine these formats in any order and can include characters such as dashes, spaces, or slashes for clarity. The Data Editor recognizes null dates and displays them as blanks. The following table lists some of the formats that you can use:

Format	Result for the Date March 22, 2005
%d-%b-%y	22-Mar-05 (default date format)
%Y-%m-%d	2005-03-22
% B-%d-%Y	March 22, 2005

Defining the Number of Decimal Places

Access the Data Editor.

To define the number of decimal places:

- Select Configure, Display.
- In the Configure Data window, in the Precision field, enter the number of decimal places that you want to use.

  The default is two decimal places.

Changing Values in Rows, Columns, or Views

Access the Data Editor.

To change values in a row, column, or view:

- Ensure that the data that you want to change is classified as dataReadWrite in the Configure Fields window.
- Select a cell in the row or column that you want to change.
- Select Edit, Fill Cells.

  In the Fill Cells window, the row and column number of the cell you selected in Step 2 appears in the Row and Column fields. If you do not select a specific cell, the default row and column number is 1.

- In the Value to Fill Cells field, enter the new value in the selected row or column.
- Select one of the following options:
  - Row.

    This option applies the change to every cell in the indicated row. You can edit this number.

  - Column.

    This option applies the changes to every cell in the indicated column. You can edit this number. This option is available only when your Data Editor displays one data value. For example, if your Data Editor displays only the global demand data, you can use this option. However, if it displays both global demand and safety stock, you cannot use this option.

  - Entire Table.

    This option applies the change to every cell in the Data Editor. This option is available only when your Data Editor displays one data value. For example, if your Data Editor displays only the global demand data, you can use this option. However, if it displays both global demand and safety stock, you cannot use this option.

- Click Apply & Close.

See Also

Customizing Views

Selecting Data Classifications and Types

The Data Editor shows data in different classifications and types. Classification makes the data editable or uneditable. Type indicates what kind of data is in the field. Ensure that the data object is set to Editable in the Properties window.

Access the Data Editor

To select data classifications and types:

- Select Configure, Fields.
- In the Configure Fields window, select a data classification:
  - A Key classification means the field is part of the primary key. You cannot edit the data in a key field.
  - A dataReadOnly classification means you can view data in this field, but you cannot edit it.
  - A dataReadWrite classification means you can view and edit the data in this field.
- Select a data type:
  - A string data type means the field contains a string of text.
  - A number data type means the field contains numbers.
  - A yesno data type means the field contains either yes or no.
  - A date data type means the field contains a date.

  If you want to use a field for period-related functions, select the field that you want in the Period section.

Creating a Period Durations File

Access the Data Editor.

To create a period durations file:

- Select Configure, Fields.
- In the Configure Fields window, click the data type button for Period and select string from the menu.
- Use a text editor to create and save the period durations file.
- In this file, enter the period information for the next few years so that you do not need to update the file each time that the period changes.

  The period durations file is a three-column table. The first column contains the name of the period, the second column contains the name of the time unit, and the third contains the number of units in the period. For example, for five periods, your file could look like this:

```
January days 31
February days 28
March days 31
April weeks 4
May weeks 4
```

- In the main Data Editor window, select Configure, Display.
- In the Configure Data window, in the Period Durations File field, enter the name of the period durations file that you created in Step 2.

See Also

Depletion

Using the #timeUnit Parameter

Using Breaks

You can use breaks to automatically calculate key fields and show one of the following values:

- Sum
- Minimum
- Maximum
- Average

All the configured break information appears in blue.

Performing Calculations with Breaks

Access the Data Editor.

To perform calculations with breaks:

- Select Configure, Breaks.
- In the Configure Breaks window, select the field where you want to perform the calculation.
- Enter a symbolic name for the break in the Name field.
- From the Break Frequency menu, select the frequency that you want the break to occur.
- From the First Break menu, select where you want the first break to occur. If you want to create a visual break following your calculation, click Put Blank After Break.
- From the Demand menu, select the calculation that you want to perform. You have the following options:

Option	Description
Nothing	Does not calculate anything (useful when you want to display a blank row)
Sum	Adds the values of this data field
Minimum	Displays the field's minimum value

Option	Description
Maximum	Displays the field's maximum value
Average	Calculates the average of the displayed values
First	Displays the first value that appears after the most recent break
Last	Displays the last value that appears before the most recent break

- Click Apply & Close.

Showing or Hiding Breaks

Access the Data Editor.

To show or hide breaks:

- Select Configure, Breaks.
- Click Actions and select either Activate All Breaks or Suspend All Breaks.

Using Expressions

You can use a variety of expressions to perform calculations on your data. This section discusses how to:

- Create fields and expressions.
- Edit expressions.
- Delete expressions.
- Recalculate expressions.
- Recalculate inverse formulas.

Creating Fields and Expressions

You can create a Data Editor field and an expression with the Expression List window and the Expression Editor.

You may want to define your own fields that give results that you need for your operations. For example, you can create fields to display the results of calculations that you performed on other fields.

Access the Data Editor.

To create fields and expressions:

- Select Configure, Expressions.
- In the Expression List window, select Actions, New.
- Select Actions, Edit.

- In the Expression Editor window, in the Field Name field, enter the name of the data field that you want to create.
- In the Formula field, enter the expression that you want to use.

  You can use the drag and drop feature to copy functions from the Functions window. From the Configure menu, select List Functions.

  If you want to use an inverse formula, enter it in the Inverse Formula field. You can use the drag and drop feature to copy functions from the Functions window. From the Configure menu, select List Functions.

- In the Expression Editor window, click Apply & Close.

  The information is saved in the Expression Editor.

  If an expression under the Formula list uses data from the Field Name list, then such data must be defined before the expression formula.

See Also

Viewing the Available Functions

Copying Expressions

Recalculating Inverse Formulas

Editing Expressions

Access the Data Editor.

To edit expressions:

- Select Configure, Expressions.
- In the Expression List window, select the expression that you want to edit.
- Select Actions, Edit.
- In the Expression Editor window, enter your changes.
- Click Apply & Close.

Deleting Expressions

Access the Data Editor.

To delete expressions:

- Select Configure, Expressions.
- In the Expression List window, select the expression you want to delete.
- Select Delete, Actions.

Recalculating Expressions

Access the Data Editor.

To recalculate expressions, select Edit, Recalculate, All Formulas.

Assigning Values to Fields

Inverse formulas use the Assign function to assign a value to a field. When you want to update the values in the assigned field, you select Recalculate and then Inverse Formulas from the edit menu. This type of inverse formula is not updated automatically.

For example, suppose a field called BlueInfo has data that is the maximum value of the fields BlueInv and BlueSafety. This calculation is specified in the Formula field of the Expression Editor.

Also, an inverse formula assigns the value from the Forecast field multiplied by two to the field named BlueDemand. This calculation is specified in the Expression Editor for the field BlueInfo.

The expressions in the Formula and Inverse Formula fields do not seem related to each other. When you use the Inverse Formula to assign a value to a field, the inverse formula does not have to correspond with the formula.

Automatically Updating the Data in a Field

An inverse formula automatically updates field values that are based on data in another location, such as another field or an expression. If you create a field based on data from another location and if you change the value of the field, the inverse formula automatically updates the values in the other location to correspond with the change.

For example, suppose you created a field called RedPercent. This field displays the percentage of the maximum of the product Red that was produced. In the Formula field, enter the formula.

Suppose the Used field displays a value of 70 and the Max field displays a value of 140. In this case, the RedPercent field displays a value of 50%. If you want the percentage to be 80% instead of 50%, you must use an inverse formula to change the values in the Used and Max fields. In the Inverse Formula field, you enter:

```
Assign(Used,RedPercent*Max.100)
```

This expression tells the Data Editor to multiply the values in the RedPercent field and the Max field, divide this number by 100, and assign it to the Used field.

If this expression is entered in the Inverse Formula field for the RedPercent field, and you change the value in that field to 80%, the Used field is updated automatically to a value of 112, which is 80% of the Max field value of 140.

Recalculating Inverse Formulas

Inverse formulas are used only in the Data Editor to:

- Assign a value to another field
- Automatically update the data in another file

Access the Data Editor.

To recalculate inverse formulas, select Edit, All Inverse Formulas, Recalculate.

You can use this option only with inverse formulas that assign a value to another field.

See Also

Recalculating Inverse Formulas

Using Expressions from Different Views

You can create a view that contains expressions that you may want to use in other views. This procedure is useful if you have the same data in several different views and you want to apply the same calculations to the data, but you want to display it in different ways.

To use expressions from a different view:

- Select File, Read Expressions.
- In the Load Expressions window, search for the file in which you want to save your data by entering the match pattern in the Filter field and click Filter.

  If the file that you want is not listed in the current directory, use the Directories field to find the directory in which the file resides. The Directories field displays the names of all your directories. Double-click the directory with the contents that you want to display in the Files list. If you click the directory name followed by /.. you move up one directory level.

- Click Only Show Views That Apply if you want to see a list of only those views that apply to the currently displayed Data Editor.

  By default, only those views with expressions that are applicable to the current data are available.

- Do one of the following:
  - Double-click a file in the Files field.

  The expressions from this file are loaded.

  - Select the view you that want from the Files field
  - Enter the file name directly into the Selection field.
- Click OK.

  The view is loaded from the file specified in the Selection field, and expressions are copied into the current view.

See Also

Using Wildcard Characters for Pattern Matching

Highlighting Expressions and Fields

You can configure the Data Editor to display data in different colors for values that violate an important boundary.

Highlighting can show:

- The results of calculations
- Specific fields

You can use the following colors:

- Black
- Blue
- Pink
- Red
- Green
- Orange
- Violet
- Yellow

For example, if you want to display a field's value in red each time that it goes over 25, and in black if it is less than or equal to 25, in the Highlight Editor, you would enter:

```
IF(Maximum>25,RED(),BLACK())
```

You can also embed highlighting expressions within each other, just as you can with regular expressions. For example, if you want to use red highlighting if the maximum value is greater than 25, the minimum value is less than 10, or you want to use blue highlighting if the minimum value is greater than 10, in the Highlight Editor, you would enter:

```
IF(Maximum>25,RED(),IF(Minimum<10,RED(),BLUE()))
```

This table shows the results, which depend on the values in the minimum and maximum fields:

Max	Min	Highlighting	Meaning
30	10	Red	The maximum is greater than 25, which matches the condition of the IF statement, so the Highlighting is red. The minimum does not need to be considered because the first condition of the IF statement was met.
9	5	Red	The maximum is less than 25, which does not match the first condition of the IF statement, so the second condition is considered. The second condition is matched, so the Highlighting is red.
12	14	Blue	Neither the first nor the second conditions are met, so the third condition is considered. It states that when the minimum is not less than 10, blue Highlighting is used.

To highlight the results of calculations:

- Create the expressions that you want with the Expressions option from the Configure menu.
- Select Configure, Highlighting.
- In the HighlightExpression List window, select the expression that you want to highlight.
- Do one of the following:

- Select Actions, Edit.
- Double-click the expression that you want to edit.

- In the Highlight Editor window, enter the color expression that specifies the conditions under which the field is to be highlighted.

  You can use the drag and drop feature to copy functions listed in the Functions window.

- Click Apply & Close.

  This information is applied, and the Highlight Editor disappears.

- In the HighlightExpression List window, click Apply & Close.

See Also

Using Wildcard Characters for Pattern Matching

Copying Expressions

To edit or delete highlighting:

You may need to edit or delete a highlighted expression. Although there is no delete function, you can remove the highlighting information for an expression by editing it.

- Select Configure, Highlighting.
- In the HighlightExpression List window, select the expression that you want to edit.
- Do one of the following:
  - In the HighlightExpression List window, select Actions, Edit.
  - Double-click the field you want to highlight.
- In the Highlight Editor window, in the Highlight Formula field, change the conditions under which you want the field to be highlighted.
- Select List Functions from the Configure menu if you want to use an expression from the Functions window. If you want to delete the Highlighting and return the field to black, replace the expression with the default BLACK().
- Click Apply & Close.

  The Highlight Editor window disappears.

- In the HighlightExpression List window, click Apply & Close.

Highlighting Specific Fields

To highlight specific fields:

You can use the HighlightExpression Editor to highlight regular data, even if it is not part of an expression or the result of a calculation.

- Select Configure, Highlighting.

- In the HighlightExpression List window, double-click the color in the Formula field.
- In the Highlight Editor window, enter the color name.
- Click Apply & Close.
- In the HighlightExpression List window, click Apply & Close.

Highlighting Expressions from Different Views

To use highlighting expressions from a different view:

- Select File, Read Highlighting.
- In the Load Highlighting window, search for the view that you want to use by entering the match pattern in the Filter field, and click Filter to activate the Filter field.

  If the view that you want is not listed in the current directory, use the Directories field to find the directory in which the view resides. The Directories field displays the names of all your directories. Double-click the directory with the contents that you want to display in the Files list.

- Select Only Show Views That Apply if you want to see a list of the views that apply only to the currently displayed Data Editor.

  By default, only those views applicable to the current data are available.

- Do one of the following:
  - Double-click the view that you want from the Files field. The view is saved in this file.
  - Select the view that you want from the Files field and it appears in the Selection field. You can also enter the file name directly in the Selection field.
- Click OK.

Viewing Available Functions

The Data Editor allows you to use many different functions to create a variety of expressions. Each function has its own components that must be written in a specific order.

To view the available functions, select Configure, List Functions.

Using Arithmetic Symbols

The functions listed in the Functions window instruct the Data Editor to perform a function, and the arithmetic symbol works within each function. The arithmetic symbols specify exactly how a function operates.

For example, the arithmetic symbols in the following IF statement are essential because they provide the conditions for the IF statement. This IF statement states that if the expression is greater than 10 but less than 20, use the red highlight. Otherwise, use the green highlight:

```
IF (expression>10 AND expression<20,RED(),GREEN())
```

The following table explains how the symbols exp1 and exp2 represent expressions:

Symbol	Explanation
Exp1 = exp2	The expressions are equal.
Exp1 != exp2	The expressions are not equal.
Exp1 < exp2	Expression1 is less than expression2.
Exp1 > exp2	Expression1 is greater than expression2.
Exp1 <= exp2	Expression1 is less than or equal to expression2.
Exp1 >= exp2	Expression1 is greater than or equal to the expression2.
Exp1 AND exp2	True, if the expressions on both sides are non-zero.
Exp1 OR exp2	True, if either expression is non-zero.
Exp1 XOR exp2	True, if one expression on either side is non-zero.
()	Left and right brackets indicate the order of operations.
+	The addition symbol.
-	The subtraction symbol.
*	The multiplication symbol.
/	The division symbol.
=	The equal symbol.
''	Use quotation marks around data field names that contain spaces, for example: *'Blue Medium'* *Note* You do not need to use quotation marks when you are using functions in Advanced Planning Agent because the database field names cannot contain spaces. However, you must use them for field names with spaces in the Data Editor.
fieldname	When you use a data field name in an expression, the value from that field is used.

Configuring Expressions in Data Editor Functions

In the Functions window, the functions are listed by name and syntax. Each function has its own syntax that you must follow. The following general rules apply to all the functions:

- If you want to use a data field name in your expressions, the value from that field is used.

- You must use single quotation marks around data field names that contain spaces. Enter them exactly as they appear: *'Storage Min'*
- Each function must be followed by a left and right parenthesis. For some functions, the parentheses remain empty, for example,*RED()*. For other functions, the parentheses must contain arguments-for example,*SUM(a,b,c)*.
- You can use multiple functions and nested functions in an expression.
- Variables in functions are indicated by *x*, *y*, or some other variable name. In place of the variable name, you can enter:
  - A number
  - A field name if the field contains a number
  - An expression that results in a number
- Variable lists in functions are indicated by *a,b...* . In place of this, you can enter:
  - A list of numbers
  - A list of fields that contain numbers
  - A list of expressions that result in a number

  For example, you could enter the SUM function, which calculates the sum of its arguments in three different ways:

  - *SUM(3,5,7)*
  - *SUM(Min,Safety,Max)*
  - *SUM(PRODUCT (Amount,Cost),Storage)*

Copying Expressions

The drag and drop feature allows you to copy text from one location and place it in another location. For example, you can pick up an expression listed in the Functions window and place it in the Expression Editor or the Highlight Editor.

To copy expressions:

1. Open the Expressions Editor window or the Highlight Editor window.
- Open the Functions window.
- In the Functions window, position the pointer on the expression that you want to use and press both mouse buttons or the middle mouse button.
- Move the icon to the field where you want to place the expression and release the mouse button.

Saving Views

After you change the Data Editor, you can save your changes as a view so that you can use them again.

When you save a view, you save the information about the following:

- How the data appears
- Printing specifications
- Expressions used to calculate or highlight data

To save a view:

- Select File, Save or Save As.
- Search for a view file where you want to save your data by entering the match pattern in the Filter field, and click Filter to activate the Filter field.

  If the view file that you want is not listed in the current directory, use the Directories field to find the directory where the file resides. The Directories field displays the names of all your directories. Double-click the directory with the contents that you want to display in the Files list.

- Click Only Show Views That Apply if you want to see a list of the views that apply to the currently displayed Data Editor.

  The default shows only those views that are applicable to the current data.

- Do one of the following:
  - Select the view that you want from the Files field and double-click it. The view is saved in this file.
  - Select the view that you want from the Files field.

  The view name appears in the Selection field. You can also enter the file name directly in the Selection field.

- Click OK.

  The view is saved in the file specified in the Selection field.

See Also

Using Wildcard Characters for Pattern Matching

Loading Views

After you have created and saved views, you can use them again. To use them again, you must load them.

To load a view:

- Select File, Load View.
- In the Open window, search for the view file that you want to load by entering the match pattern in the Filter field, and click Filter to activate the Filter field.
- If the view file that you want is not listed in the current directory, use the Directories field to find the directory in which the file resides.

- Click Only Show Views That Apply if you want to see a list of the views that apply to the currently displayed Data Editor only.

  The default shows only those views that are applicable to the current data.

- Do one of the following:
  - Select the view that you want from the Files field and double-click it. This file is loaded, and the view appears in the Data Editor.
  - Select the view that you want from the Files field, and it appears in the Selection field. You can also enter the file name directly in the Selection field.
- Click OK.

See Also

Using Wildcard Characters for Pattern Matching

Deleting Views

To delete a view:

- Select File, Delete.
- In the Delete window, search for the view file that you want to delete by entering the match pattern in the Filter field and click Filter to activate the Filter field.

  If the view file you want is not listed in the current directory, use the Directories field to find the directory in which the file resides. Click Only Show Views That Apply if you want to see a list of the views that apply to the currently displayed Data Editor only. By default, only those views that are applicable to the current data are available.

- Do one of the following:
- Select a view in the Files field, and double-click it.
  - Select a view in the Files field and it appears in the Selection field.
  - Enter the file name directly in the Selection field.
- Click OK.
- In the confirmation window, click OK.

Printing Reports

You can print your report in a variety of ways by changing the appearance of the Data Editor through views and by setting the print options.

To print reports:

- Select File, Print.

Note

If you are using Windows NT, the Printer Command field is called Printer.

- In the Print window, in the Title field, enter the title that you want to appear on each page.
- In the Footer field, enter the footer that you want to appear on each page.
- Select Repeat Row Headings if you want the row headings to appear on every page.
- Select Repeat Column Headings if you want the column headings to appear on every page.
- Select Print All Pages if you want to print all the Z dimension layers, even if they are not currently displayed.
- In the Printer field, select the type of output that you want.

  Postscript outputs a file with full formatting, including a grid, clear fonts, and footers. ASCII outputs a file with basic formatting in courier. HTML produces output that can be browsed with Netscape or other browsers on the World Wide Web.

- (Optional.) If you select Postscript, do the following:
  - In the Paper Size field, select the size of paper that you use.
  - In the Orientation field, select Landscape or Portrait.
  - If you want lines to appear between your rows and columns, select Show Grid.
  - Select one of the following options:

Option	Description
Fit on One Page	Prints all the information on one page. If your data can fill more than one page, the font size is reduced so that all your data fits onto one page.
Fit Width	For a wide report that you do not want to condense to one page, you can specify the width of the report in pages.
Fit Height	For a long report that you do not want to condense to one page, you can specify the number of pages for the report.
Use Font Size	Specify a font size that you want in your printed report.

  - On UNIX, in the Printer Command field, you can specify the print command or use the default, which is */usr/bin/lp -c* .
    On Windows NT, you can specify the printer or use the default, which is >*prn.* On both UNIX and Windows NT, you can redirect the data to a file instead of a printer. To redirect to a file, enter > < *filename* >.
  - Click Apply & Print.
- If you select ASCII:
  - Specify the number of characters wide in the Page Width field.
  - Specify the number of characters high in the Page Height field.
  - On UNIX, in the Printer Command field, you can specify the print command or use the default, which is */usr/bin/lp -c* .
    On Windows NT, you can specify the printer or use the default, which is >*prn* . On both UNIX and Windows NT you can redirect the data to a file instead of a printer. To redirect to a file, enter > < *filename* >.

- Click Apply & Print.

- If you select HTML:
  - The Add HTML Header option adds the HTML codes to the output file so that it can be viewed as a stand-alone Web page. If you do not select this option, an output file is produced that is suitable for inclusion in other Web pages.
  - The Cell Border option adds borders to the cells in the tables in the output file. If you do not select this option, the tables have no lines between their cells.
  - In the Printer field, you can redirect the print destination or use the default file destination, which is $Home/<table name>.html.
  - Click Apply & Print. The file is produced and can be viewed in your Web browser.

Saving to Database Tables

You can save your data to a specific table in your database using the Save Base Data option from the File menu. The data originates in a database table, and when you use the Save Base Data option, you save to that same database table. You cannot save to a different database table.

To save to a database table:

- Select File, Save Base Data.

  A confirmation window appears. Ensure that the table specified in the confirmation window is the one to which you want to save your changes.

- Click OK.

Exporting Data to Files

You have two options when you export formatted data to a file with a slight difference between them:

- To export to a file to which you have exported previously, select File, Export.
- To export a view for the first time or to change the information about how you export a view, select File, Export As. When you select this option, the Export As window appears.

Access the Data Editor.

To export data to a file from the Data Editor:

1. From the File menu, select Export or Export As.

   If you select Export As, the Export As window appears.

2. (Optional.) To search for the file to which you want to export your data, enter the match pattern in the Filter field and click the Filter button, which activates the Filter field.

   If the file that you want is not listed in the current directory, use the Directories field to find the directory in which the file resides.

3. Select Put All Strings in Quotes (Lotus Import Format) if you want your data to be put in quotation marks.

   Lotus recognizes this format, so if you plan to export to Lotus, you must select this option.

4. Do one of the following:
   - Select a file in the Files field and double-click it. The data is exported to this file.
   - Select a file in the Files field and it appears in the Selection field. You can also enter the file name directly in the Selection field.
5. Click OK.

Using the Inverse Formula to Automatically Update Data

An inverse formula automatically updates field values that are based on data in another location, such as another field or an expression. If you create a field based on data from another location and change the value of the field, the inverse formula automatically updates the values in the other location to reflect the change.

For example, if you created a field called RedPercent, the percentage of the maximum of the product Red that was produced is displayed. In the Formula field, enter the following formula:

```
Percent(Used,Max)
```

For example, suppose the Used field displays a value of 70 and the Max field displays a value of 140. In this case, the RedPercent field displays a value of 50 percent. If you want the percentage to be 80 percent instead of 50 percent, you must use an inverse formula to change the values in the Used and Max fields. In the Inverse Formula field, you enter:

```
Assign(#Used,RedPercent * Max/100)
```

This expression tells the Data Editor to multiply the values in the RedPercent field and the Max field, divide the resulting number by 100, and assign the final number to the Used field.

If the expression is entered in the Inverse Formula field for the RedPercent field and you change the value in Inverse Formula field to 80 percent, the Used field is updated to a value of 112, which is 80 percent of the Max field value of 140.

Understanding Expressions and Functions

You can use expressions and functions in Advanced Planning Agent and the Data Editor to find, calculate, and change data. This section provides overviews of Advanced Planning Agent and Data Editor functions, date functions, color functions, trigonometric functions, rolling functions, string functions, and provides a function reference.

Advanced Planning Agent and Data Editor Functions

This table lists all of the functions and indicates whether you can use them in Advanced Planning Agent expressions or in the Data Editor.

Function	Advanced Planning Agent	Data Editor	Description
ABS(*x*)	X	X	Returns the absolute value of an expression.
ABSDATE	X		Returns a localized date and time.
ACOS(*x*)	X		Returns the arc cosine of x.
ADVANCEDAYS (*date,# of days*)	X		Changes a date by a specified number of days.
AND,OR, XOR	X	X	Boolean operators used in IF functions.
ASCIICONVERT(*string*)	X		Converts a string to ASCII characters.
ASIN(*x*)	X		Returns the arc sine of x.
ASSIGN*(#tofield,expr*)		X	Assigns the result of the expression to a specific field.
ATAN (*x*)	X		Returns the arc tangent of x.
ATAN2(*x,y*)	X		Returns the arc tangent of x ÷ y.
AVG(*a,b,...*)	X	X	Returns the average of a list of values.
AVG(*num_expression*)	X		Returns the average of a field in the aggregation filter.
BACKGROUND()		X	Changes the background and foreground to the same color.
BETWEEN(*lb,y,ub*)	X	X	Returns 1 if y >= lb and y <= ub ; otherwise returns 0.
BLACK()		X	Changes color to black.

Function	Advanced Planning Agent	Data Editor	Description
BLUE()		X	Changes color to blue.
BUCKETLENGTHS	X		Bucketizes length of time (has a start date and a length of time).
BUCKETPERIODS	X		Bucketizes increments of time.
BUCKETVALUES	X		Specifies the exact boundaries of buckets.
COS(*x*)	X		Returns the cosine of x.
COSH(*x*)	X		Returns the hyperbolic cosine of x.
COVER(*unit,d,c,*[*last/avg*])		X	Returns the amount of time that your inventory lasts.
DATETOUMTSEC (*date*) : integer	X		Returns the Universal Mean Time.
DAY (*date*)	X		Returns the day of the month (a number from 1 to 31) of the specified date.
DAYDUR (*numDays*)	X		Returns the number of seconds in the specified number of days.
Depletion(*d,o,e,t,f/b,l/a*)		X	Determines the ability to meet future demand based on inventory levels.
EQTOL(*x,y,tol*)	X	X	Returns 1 if the difference between x and y = tol.
ERROR(*#errorMessage*)		X	Displays an error message.
EXP(*x*)	X	X	Returns e to the power of x.
FALSE()	X	X	Returns 0 (does not take an argument).
FINDNEXTDAY(*date, day,option*)	X		Returns date for the next occurrence of day following date.
GAMMA (*x*)	X		Returns the factorial of x -1.
GETDATE (*str,format*)	X		Converts a string to a date.
GETDAYSINMONTH (*date*)	X		Returns the number of days in a specific month.
GETENV(*string*)	X		Returns the character value of an environment string.
GETOL(*x,y,tol*)	X	X	Returns 1 if the difference between x and y is >= the tolerance (tol).
GREEN()		X	Changes color to green.

Function	Advanced Planning Agent	Data Editor	Description
GTTOL(*x,y,tol*)	X	X	Returns 1 if the difference between x and y is > the tolerance (tol).
HOUR (*date*)	X		Returns the hour of the day (a number from 0 to 23) of the specified date.
HOURDUR (*numHours*)	X		Returns the number of seconds in the specified number of hours.
IF(*cond,x,y*)	X	X	Returns x if the condition is true; y if the condition is false.
INDEX (*string,pattern,count*)	X		Searches for the count occurrence of pattern in string and returns the index of the start of the occurrence.
INFINITESIMAL()	X		Returns the value 10-30.
INFINITY ()	X		Returns the value 10+30.
INSERT (*string,index,insertString*)	X		Inserts into string at index and returns new string.
INT(*x*)	X	X	Returns the integer value of a value; truncates decimal point values.
InvCover (*time,d,c,*[*last/avg*])		X	Returns amount of inventory required to meet demand in a time period.
InverseDepletion(*d,o,e,t,f/b,l/a*)		X	Edits the value calculated by the depletion function and recalculates.
ISNULL(*expr*)	X		Returns 1 if the expr is null; otherwise, returns 0.
KeepBlanks(;;;*kb format*)	X		Keeps trailing blanks.
KillBlanks (*string*)	X		Removes the blanks at the end of a string.
LETOL(*x,y,tol*)	X	X	Returns TRUE() if the difference between x and y is <= tolerance (tol).
LN(*x*)	X	X	Returns the natural logarithm (base e) of value x.
LOG(*x*)	X	X	Returns the common logarithm (base 10) of value x.
LTTOL (*x,y,tol*)	X	X	Returns TRUE() if the difference between x and y is < tolerance (tol).
MAKEDATE (*y,m,d,h,m,s*)	X		Returns the date specified.

Function	Advanced Planning Agent	Data Editor	Description
MAKEDATEFROMWEEK (*y,w,n*)	X		Returns a date from a week number.
MAKESYSKEY (*str*)	X		Creates a system key that is represented as a string.
MAX(*a,b,...*)	X	X	Returns the maximum value.
MAX(*expr*)	X		Returns the maximum value, in an aggregation filter.
MIN(*a,b,...*)	X	X	Returns the minimum value.
MIN(*expr*)	X		Returns the minimum value, in an aggregation filter.
MINUTE (*date*)	X		Returns the minute of the hour (a number from 0 to 59) of the specified date.
MINUTEDUR (*numMinute*)	X		Returns the number of seconds in the specified number of minutes.
MONTH (*date*)	X		Returns the month of the year (a number from 1 to 12) of the specified date.
NEG(*x*)	X	X	Returns 1 (true) if x is negative.
NETOL(*x,y,tol*)		X	Returns TRUE() if the difference between x and y is <> tolerance (tol).
NEWSYSKEY()	X		Creates a unique system key of type syskey.
Next (*#field,value*)		X	Returns the value of the next field or the value that you specify as the value.
NEXTMONTH (*date*,[*months*])	X		Returns the first day of month a number of months in the future from date.
NEXTYEAR (*date*[,*years*])	X		Returns the first of the year that is a number of years in the future from date.
NOT(*x*)	X	X	Returns 1 (true) if x is 0.
NOW()	X		Returns the current date.
NULL (*expr*)	X		Returns a null value of the same data type as the expr.
NULLDEF (*expr1 , expr2*)	X		If expr1 is null, returns expr2; otherwise, returns expr1.
ORANGE()		X	Changes color to orange.

Function	Advanced Planning Agent	Data Editor	Description
OVERUNDER(*lb,x,ub*)	X	X	Returns 0 if x is between lb and ub ; otherwise, returns 1.
PARSESTRING (*string,delim,n*)	X		Splits a string into parts separated by the delimiter and returns the nth part.
PERCENT(*x,y*)	X	X	Returns the value of; if x =0 or y =0, returns 0.
PI()	X	X	Returns the value 3.1415926.
PINK()		X	Changes color to pink.
POS(*x*)	X	X	Returns 1 if x is greater than 0.
POWER(*x,y*)	X	X	Returns x to the power of y.
Prev(*field,value*)		X	Returns the value of the previous field or the value that you specify as the value.
PRODUCT(*a,b,...*)	X	X	Multiplies the list of numbers in parentheses.
PRODUCT(*num_expr*)	X		Returns the product of values in an aggregated field.
R_AVG (*num_expr,range*)	X		Returns the rolling average (used in the aggregation filter).
R_COVER (*x,y*[,*flag*][,*period*],*units*])	X		Returns the periods of coverage for an inventory level (used in the aggregation filter).
R_MAX (*expression,range*)	X		Returns the rolling maximum (used in the aggregation filter).
R_MIN (*expression,range*)	X		Returns the rolling minimum (used in the aggregation filter).
R_PREV (*expr,default,offset*)	X		Returns a previous value; otherwise, returns the default. (used in the aggregation filter).
R_PRODUCT (*num_expr,range*)	X		Returns the rolling product (used in the aggregation filter).
R_SUM (*num_expr,range*)	X		Returns the rolling sum (used in the aggregation filter).
RAND(*x,y*)	X	X	Returns a random number between x and y.

Function	Advanced Planning Agent	Data Editor	Description
RED()		X	Changes color to red.
REGEXPR (*string,pattern*)	X		Creates a substring by matching on a string.
R elOffset (*field,offset,value*)		X	Returns the value from a relative position or the value.
REMAINDER (*x,y*)	X		Returns the remainder of x divided by y.
REMOVE (*string,index,length*)	X		Removes length characters from string, starting at index, and returns new string.
REPLACEINDEX (*string,index,length, replaceString*)	X		Replaces characters in string, starting at index, and returns new string.
REPLACEPATTERN (*string,pattern, replaceString*)	X		Searches for first occurrence of pattern in string, replaces the occurrence with replacestring, and returns new string.
ROUND(*x,number*)	X	X	Rounds the expression x to the indicated number of decimal places.
RTRIMBLANKS(*string*)	X		Removes the blanks at the end of a string.
R_AVG (*num_expr,range*)	X		Returns the rolling average (used in the aggregation filter).
R_COVER (*x,y*[,*flag*][,*period*],*units*])	X		Returns the periods of coverage for an inventory level (used in the aggregation filter).
R_MAX (*expression,range*)	X		Returns the rolling maximum (used in the aggregation filter).
R_MIN (*expression,range*)	X		Returns the rolling minimum (used in the aggregation filter).
R_PREV (*expr,default,offset*)	X		Returns a previous value; otherwise, returns the default (used in the aggregation filter).
R_PRODUCT (*num_expr,range*)	X		Returns the rolling product (used in the aggregation filter).
R_SUM (*num_expr,range*)	X		Returns the rolling sum (used in the aggregation filter).
SECOND (*date*)	X		Returns the second of the hour (a number from 0 to 59) of the specified date.
SECONDDUR (*numSeconds*)	X		Returns the number of seconds.

Function	Advanced Planning Agent	Data Editor	Description
SEQUENCE(*a,b,...*)		X	Ensures that the values or expressions are performed in order.
SIGN(*x*)	X	X	Returns -1 if x is less than 0; 0 if x equals 0; 1 if x is greater than 0.
SIN(*x*)	X		Returns the sine of x.
SINH(*x*)	X		Returns the hyperbolic sine of x.
SQRT(*x*)	X	X	Returns the positive square root of x.
STDDEV (*a,b,...*)	X	X	Returns the standard deviation of a list of values.
STDDEV (*num_expr*)	X		Returns the standard deviation of values in a field.
STRINGCAST (*number*/*date*)	X		Converts a number or a date to a string.
STRINGCONCAT (*str1*,*str2*,...)	X		Concatenates the strings in the order that they are listed.
STRINGLENGTH (*string*)	X		Returns the number of character in a string.
STRINGTODOUBLE (*string*)	X		Converts a numeric string from character to double.
STRINGTOINT (*string*)	X		Converts a numeric string from character to integer.
SUBSTRING (s*tring*,*x*,*y*)	X		Returns the substring of length y , beginning at the x th character in the string.
SUM(*a,b,...*)	X	X	Returns the sum of a list of values or expressions.
SUM(*num_expr*)	X		Returns the sum of values in an aggregated field.
TAN (*x*)	X		Returns the tangent of x.
TANH(*x*)	X		Returns the hyperbolic tangent of x.
TOLOWER (*string*)	X		Converts all alphabetic characters to lower case and returns new string.
TOUPPER (*string*)	X		Converts all alphabetic characters to upper case and returns new string.

Function	Advanced Planning Agent	Data Editor	Description
TRUE( )	X	X	Returns 1 (does not take an argument).
VIOLET ()		X	Changes color to violet.
WEEKDAY (*date*)	X		Returns the day of the week as an integer.
WEEKDUR (*numWeeks*)	X		Returns the number of seconds in the specified number of weeks.
WEEKNUMBER (*date*,*num*)	X		Returns the number of the week.
YEAR (*date*)	X		Returns the year of the specified date.
YEARDAYNUMBER(*date*)	X		Returns the day-number, where January 1 is day one.
YELLOW()		X	Changes color to yellow.
ZERO(*x*)	X	X	Returns 1 if x equals 0.

Date Functions

Date functions are available only in Advanced Planning Agent. They are useful for doing date comparisons, date manipulation, and date arithmetic.

The date functions take regional differences into account if your UNIX time management feature is set up correctly.

Note

The IBM AIX system does not support dates or times prior to December 31, 1969.

Date Function	Description
DAY(*date*)	Returns the day of the month (a number from 1 to 31) of the specified date.
DAYDUR(*numDays*)	Returns the number of seconds in the specified number of days.
FINDNEXTDAY(*start_date, weekDayNum*, TRUE()|FALSE())	Returns the date for the next occurrence of the weekday specified by the weekDayNum and following the start_date.
GETDATE(*str*,*format*)	Converts a string to a date.

Date Function	Description
GETDAYSINMONTH(*date*)	Returns the number of days in a specific month.
HOUR(*date*)	Returns the hour of the day (a number from 0 to 23) of the specified date.
HOURDUR(*numHours*)	Returns the number of seconds in the specified number of hours.
MAKEDATE(*yy,mm,dd,hh,mm,ss*)	Returns the date specified.
MAKEDATEFROMWEEK(*year,week,num*)	Returns a date from a week number.
MINUTE(*date*)	Returns the minute of the hour (a number from 0 to 59) of the specified date.
MINUTEDUR(*numMinute*)	Returns the number of seconds in the specified number of minutes.
MONTH(*date*)	Returns the month of the year (a number from 1 to 12) of the specified date.
NEXTMONTH(*date*,[*number months*])	Returns the first day of the month that is a number of months in the future from the specified date.
NEXTYEAR(*date*[,*number Years*])	Returns the first of the year that is a number of years in the future from the specified date.
NOW()	Returns the current date.
SECOND(*date*)	Returns the second of the hour (a number from 0 to 59) of the specified date.
SECONDDUR(*numSeconds*)	Returns the number of seconds.
WEEKDAY(*date*)	Returns the day of the week as an integer.
WEEKDUR(*numWeeks*)	Returns the number of seconds in the specified number of weeks.
WEEKNUMBER(*date*,*num*)	Returns the number of the week.
YEAR(*date*)	Returns the year of the specified date.
YEARDAYNUMBER(*date*)	Returns the day-number, where January 1 is day one.

Color Functions

The color functions are available only in the Data Editor.

Color Function	Description
BACKGROUND()	Changes the background and foreground to the same color.
BLACK()	Changes color to black.
BLUE()	Changes color to blue.
GREEN()	Changes color to green.
ORANGE()	Changes color to orange.
PINK()	Changes color to pink.
RED()	Changes color to red.
VIOLET()	Changes color to violet.
YELLOW()	Changes color to yellow.

Trigonometric Functions

Trigonometric functions are available only in Advanced Planning Agent. You can use the trigonometric functions to help calculate costs based on geographic distances.

Trigonometric Function	Description
ACOS(*x*)	Returns the arc cosine of x
ASIN(*x*)	Returns the arc sine of x
ATAN(*x*)	Returns the arc tangent of x
ATAN2*(x,y*)	Returns the arc tangent of x ÷ y
COS(*x*)	Returns the cosine of x
COSH(*x*)	Returns the hyperbolic cosine of x
SIN(*x*)	Returns the sine of x
SINH(*x*)	Returns the hyperbolic sine of x
TAN*(x*)	Returns the tangent of x
TANH(*x*)	Returns the hyperbolic tangent of x

Rolling Functions

You use rolling functions only in the Aggregation filter.

Rolling Function	Description
R_AVG(*numeric expression,range*)	Returns the rolling average.
R_COVER(*x,y*[,*flag*][,*period*][,*units*])	Returns the periods of coverage for an inventory level.
R_MAX(*expression,range*)	Returns the rolling maximum.
R_MIN(*expression,range*)	Returns the rolling minimum.
R_PREV(*expression,default,offset*)	Returns a previous value, depending on the offset; if no such value exists, returns the default.
R_PRODUCT(*numeric_expression,range*)	Returns the rolling product.
R_SUM(*numeric_expression,range*)	Returns the rolling sum.

String Functions

String functions are available only in Advanced Planning Agent for manipulation of character strings.

String Function	Description
ASCIICONVERT(*string*)	Converts a string to ASCII characters.
INDEX (*string,pattern,count*)	Searches for the count occurrences of pattern in a string.
INSERT (*string,index,insertString*)	Inserts insertString into a string at index.
KillBlanks (*string*)	Removes the blanks at the end of a string.
PARSESTRING(*string,delimiter,n*)	Splits a string into parts separated by the delimiter and returns the nth part.
REGEXPR(*string,pattern*)	Creates a substring by matching on a string.
REMOVE (*string,index,length*)	Removes length characters from a string.
REPLACEINDEX (*string,index,length,replaceString*)	Replaces length characters in a string.

String Function	Description
REPLACEPATTERN (*string,pattern,replaceString*)	Replaces the first occurrence of pattern in a string with replacesString.
RTRIMBLANKS(*string*)	Removes the blanks at the end of a string.
STRINGCAST(*number/date*)	Converts a number or a date to a string.
STRINGCONCAT(*string1,string2,...*)	Concatenates the strings in the order that they are listed.
STRINGLENGTH(*string*)	Returns the number of character in a string.
STRINGTODOUBLE(*string*)	Converts a numeric string from character to double.
STRINGTOINT(*strin*g)	Converts a numeric string from character to integer.
SUBSTRING(*string,x,y*)	Returns the substring of length y , beginning at the x th character in the string.
TOLOWER(*string*)	Converts all alphabetic characters to lower case.
TOUPPER(*string*)	Converts all alphabetic characters to upper case.

Function Reference

This section discusses how to use the following Advanced Planning Agent and Data Editor functions:

- ABS
- ABSDATE
- ACOS
- ADVANCEDAYS
- AND,OR, and XOR
- ASCIICONVERT
- ASIN
- ASSIGN
- ATAN
- ATAN2
- AVG
- BACKGROUND
- BETWEEN

- BLACK
- BLUE
- BUCKETLENGTHS
- BUCKETPERIODS
- BUCKETVALUES
- COS
- COSH
- COVER
- DATETOUMTSEC
- DAY
- DAYDUR
- Depletion
- EQTOL
- ERROR
- EXP
- FALSE
- FINDNEXTDAY
- GAMMA
- GETDATE
- GETDAYSINMONTH
- GETENV
- GETOL
- GREEN
- GTTOL
- HOUR
- HOURDUR
- IF
- INDEX
- INFINITESIMAL
- INFINITY
- INSERT
- INT
- InvCover
- InverseDepletion

- ISNULL
- KeepBlanks
- KillBlanks
- LETOL
- LN
- LOG
- LTRIMBLANKS
- LTTOL
- MAKEDATE
- MAKEDATEFROMWEEK
- MAKESYSKEY
- MAX
- MIN
- MINUTE
- MINUTEDUR
- MONTH
- NEG
- NETOL
- Next
- NEXTMONTH
- NEXTYEAR
- NEWSYSKEY
- NOT
- NOW
- NULL
- NULLDEF
- ORANGE
- OVERUNDER
- PARSESTRING
- PERCENT
- PI
- PINK
- POS
- POWER

- Prev
- PRODUCT
- R_AVG
- R_COVER
- R_MAX
- R_MIN
- R_PREV
- R_PRODUCT
- R_SUM
- RAND
- RED
- REGEXPR
- RelOffset
- REMAINDER
- REMOVE
- REPLACEINDEX
- REPLACEPATTERN
- ROUND
- RTRIMBLANKS
- SECOND
- SECONDDUR
- SEQUENCE
- SIGN
- SIN
- SQRT
- STDDEV
- STRINGCAST
- STRINGCONCAT
- STRINGLENGTH
- STRINGTODOUBLE
- STRINGTOINT
- SUBSTRING
- SUM
- TAN

- TANH
- TOLOWER
- TOUPPER
- TRIMBLANKS
- TRUE
- VIOLET
- WEEKDAY
- WEEKDUR
- WEEKNUMBER
- YEAR
- YEARDAYNUMBER
- YELLOW
- ZERO

ABS

Syntax

```
ABS (x)
```

Description

The ABS function returns the absolute (non-negative) value of an expression. In Advanced Planning Agent, if the value is null, the function returns null.

Parameters

Parameter	Description
X	Value or expression for which you want to calculate the absolute value.

Example

You can use this function, for example, if you want to calculate the difference between a forecast value and an actual value, but you do not want the calculation to result in a negative number. For another example, demand values in Production Scheduling-Process are negative and demand may not be stored as negative elsewhere, so you may need to convert negative values to positive values.

In the Data Editor, if you want to ensure that the result of the sum of the two fields *hours available* and *hours used* is a positive number, enter:

```
ABS(SUM( 'hours available','hours used') )
```

In Advanced Planning Agent, if you want to ensure that the result of the sum of two fields is a positive number, enter:

```
ABS(SUM($1. ColumnA ,$1. ColumnB ))
```

ACOS

Syntax

```
ACOS (x)
```

Description

The ABS function returns the absolute (non-negative) value of an expression.

In Advanced Planning Agent, if the value is null, the function returns null.

Parameters

Parameter	Description
X	Value or expression for which you want to calculate the absolute value.

AND, OR, XOR Comparisons

You use these comparisons as you would other comparisons in Advanced Planning Agent expressions, such as = , <= , and >= . In Boolean comparisons, true is equivalent to 1, and false is equivalent to 0.

The following tables show the results when you compare two expressions:

AND		
Expression 1	**Expression 2**	**Result**
0	0	0
0	1	0
1	0	0
1	1	1
If both expressions are true, the result is true.		

OR		
Expression 1	**Expression 2**	**Result**
0	0	0
0	1	1
1	0	1
1	1	1
If either expression is true or if both expressions are true, then the result is true.		

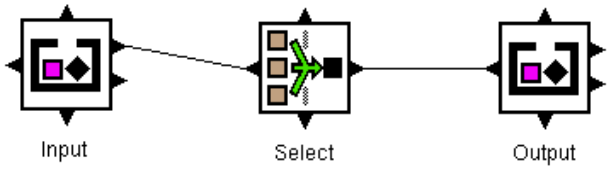
XOR		
Expression 1	**Expression 2**	**Result**
0	0	0
0	1	1
1	0	1
1	1	0
If only one expression is true, the result is true.		

For another example, suppose you have the following data table:

Product	Demand
Red	50
Blue	100
Green	75
Red	25
Blue	150

From the table, you want to select the rows in which the demand for product Red is greater than 30, and the rows in which the demand for product Blue is greater than 125. You want to send this data to another data table.

You can accomplish this task using a selection filter that has an expression that uses both the *OR* and *AND* comparisons. Your data flow would be similar to the following:



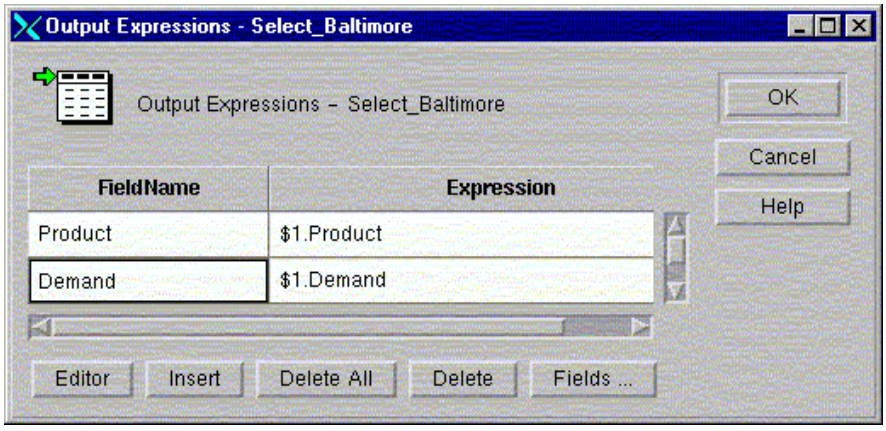
Data flow with selection filter

The selection filter uses the following selection criteria:

($1.Product="Red" AND $1.Demand > 30) OR
($1.Product="Blue" AND $1.Demand > 125)

The expression is too long to display in the Input Selection Criteria window, so you can display it in its entirety in the Expression Editor.

The output expressions in this example are shown in the following graphic. The output table uses the same field names as the input table:



Output expressions

After the data flow runs, the output table is as follows:

Product	Demand
Blue	150
Red	50

Only two rows met the criteria to be selected, and those are the two rows that the output table contains.

ABSDATE

Syntax

```
ABSDATE (integer): date
```

Description

The ABSDATE function takes the number of seconds elapsed since Jan 1 1970 00:00:00 UMT (Universal Mean Time, also called Greenwich Mean Time) and returns a localized date and time. This function is the inverse of the DATETOUMTSEC function.

See Also

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

ADVANCEDAYS

Syntax

```
ADVANCEDAYS ( date,# of days )
```

Description

The ADVANCEDAYS function, available in Advanced Planning Agent only, changes a date by a specified number of days. You specify a positive number to move the date forward or a negative number to move the date backward. If the value is null, the function returns null.

The ADVANCEDAYS function replaces the NEXTWEEKDAY function.

Example

For example, the following expression moves a date ahead by one week:

```
ADVANCEDAYS($1.Date,7)
```

See Also

ABSDATE

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

ASCIICONVERT

Syntax

```
ASCIICONVERT ( string)
```

Description

The ASCIICONVERT function, available in Advanced Planning Agent only, converts a string to ASCII characters. Only the parts of the string that match one of the required formats are converted.

Format	Meaning
\0xhh	The hh is a hexadecimal number from 00 to ff.
\0ooo	The ooo is an octal number from 000 to 377.
\ ddd	The ddd is a decimal number from 0 to 255.
\a	Alert
\b	Backspace
\f	Form-feed
\n	New-line
\r	Carriage return
\t	Tab

\v	Vertical tab
\\	Backslash

Example

For example, the hexadecimal equivalent of the pound sign (#) is 23. Therefore, the following function:

```
ASCIICONVERT("This is the pound sign: \0x23")
```

returns

This is the pound sign: #

Parts of the string that do not match one of the formats are not converted. For example, *\k* simply returns *k* . If a value is null, the function returns null.

For a list of numbers and their corresponding ASCII characters, see UNIX help.

ASIN

Syntax

```
ASIN ( x )
```

Description

The ASIN function, available in Advanced Planning Agent only, returns sin-1(*x*), the arc sine of *x* , where *x* is measured in radians. If the value is null, the function returns null.

See Also

ACOS

ATAN

ATAN2

COS

COSH

SIN

SINH

TAN

TANH

ASSIGN

Syntax

```
ASSIGN ( #tofield,expr )
```

Description

The ASSIGN function is available in the Data Editor only. It takes the result of the expression (*expr*) and places it in the specific field (*#tofield*). This function is used only with inverse formulas.

Enter the number sign (#) before the name of the field to which you are assigning the data.

Examples

For example, if you want to calculate the sum of the fields YellowBig and RedBig, and apply it to the field BigTotal, enter:

```
ASSIGN(#BigTotal,SUM(YellowBig,RedBig))
```

If you want to change the value of several fields when you edit a single expression, you multiply several assign statements in the inverse formula. To perform three Assign functions when you edit the field called InputField, enter:

```
ASSIGN(#FieldA,InputField/1000) * ASSIGN(#FieldB,InputField*2) *
Assign(#FieldC,InputField)
```

See Also

- *Alphabetical List of Functions*

ATAN

Syntax

```
ATAN( x )
```

Description

The ATAN function, available in Advanced Planning Agent only, returns tan-1(*x*), the arc tangent of *x* , where *x* is measured in radians. If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN2

COS

COSH

SIN

SINH

TAN

TANH

ATAN2

Syntax

```
ATAN2( x,y )
```

Description

The ATAN2 function, available in Advanced Planning Agent only, returns tan-1(*xy*), where *x* and *y* are measured in radians. If either value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

COS

COSH

SIN

SINH

TAN

TANH

AVG

Syntax

```
AVG( a,b,... )
AVG( $1.a,$1.b..... ) calculates the average of the fields, (
a,b,...).
```

Description

The AVG function finds the average of the listed values. To use this function in an aggregation filter, specify a single value representing the field name. If a value is null, it is ignored.

AVG(*$1.a*) is used in the aggregation filter to calculate the average of a field in all records with the same aggregation key.

Examples

If you want a null value to be considered as a 0, use the NULLDEF function with the AVG function, For example, if you want the average of all fields, including fields with NULL values, enter:

```
AVG(NULLDEF($1. field ,0)) .
```

In Advanced Planning Agent, if you want to find the average of three columns in two tables, enter:

```
AVG($1.ColumnA,$2.ColumnA,$2.ColumnB)
```

In the Data Editor, if you want to find the average of the PctUsed field for BigRed, BigBlue, and BigYellow, enter:

```
AVG('PctUsed_BigRed','PctUsed_BigBlue','PctUsed_BigYellow')
```

BACKGROUND

Syntax

```
BACKGROUND()
```

Description

The BACKGROUND color function is available in the Data Editor only. This function changes the background and foreground to the same color. You will not be able to see your data if you use this option. It can be used to hide certain values.

Example

If you want to display only the values in the PctUsed field that are between the values of 1 and 9, you can use the IF, BETWEEN and BACKGROUND functions. Enter the following expression in the HighlightExpression Editor:

```
IF(BETWEEN(1,PctUsed,9),BLACK(),BACKGROUND())
```

See Also

BLACK

BLUE

GREEN

ORANGE

PINK

RED

VIOLET

YELLOW

BETWEEN

Syntax

```
BETWEEN( lb,y,ub )
```

Description

The BETWEEN function returns TRUE() if the value of the number is equal to or greater than the lower boundary (*lb*) and equal to or less than the upper boundary (*ub*). If the number is not between the upper and lower boundary, the result is FALSE(). If a value is null, the function returns null.

Example

In the Data Editor, if you want to ensure that your value for a field called Production is between your minimum of 10 and maximum of 25, enter:

```
BETWEEN(10,Production,25)
```

In Advanced Planning Agent expressions, if you want to ensure that the output values for a field called Production are between 10 and 25, inclusive, enter:

```
BETWEEN(10,$1.Production,25)
```

See Also

AND,OR, and XOR

FALSE

IF

TRUE

BLACK

Syntax

```
BLACK()
```

Description

The BLACK function is available in the Data Editor only. This function changes the color of a value to black.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

See Also

BACKGROUND

BACKGROUND

BLUE

GREEN

ORANGE

PINK

RED

VIOLET

YELLOW

BLUE

Syntax

```
BLUE()
```

Description

The BLUE function is available in the Data Editor only. This function changes the color of a value to blue.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

If you want the value of a field to be displayed in blue if the inventory level goes below the safety level, enter:

```
IF('Inventory Level'<'Safety Level',BLUE(),BLACK())
```

See Also

BACKGROUND

BLACK

GREEN

ORANGE

PINK

RED

VIOLET

YELLOW

BUCKETLENGTHS

Syntax

```
BUCKETLENGTHS( $1.eventField, startValue, bucketLength,
bucket2Length,... )
```

Description

The BUCKETLENGTHS function, available only in Advanced Planning Agent, has a start date and a length of time-for example, a start data of January 1 plus 30 days.

See Also

BUCKETPERIODS

BUCKETVALUES

BUCKETPERIODS

Syntax

```
BUCKETPERIODS( $1.dateField, "f"/"b", startDate, bucketLength,
#buckets,... )
```

Description

The BUCKETPERIODS function, available only in Advanced Planning Agent, bucketizes increments of time.

See Also

BUCKETLENGTHS

BUCKETVALUES

BUCKETVALUES

Syntax

```
BUCKETVALUES ( $1.eventField, bucket1Start, bucket2Start,.. bucketEnd
)
```

Description

The BUCKETVALUES function, available only in Advanced Planning Agent, specifies the exact boundaries of the buckets.

See Also

BUCKETLENGTHS

BUCKETPERIODS

COS

Syntax

```
COS ( x )
```

Description

The COS function is available in Advanced Planning Agent only. It returns the cosine of x . If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

ATAN2

COSH

SIN

SINH

TAN

TANH

COSH

Syntax

```
COSH ( x )
```

Description

The COSH function, available only in Advanced Planning Agent, returns the hyperbolic cosine of *x* . If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

ATAN2

COS

SIN

SINH

TAN

TANH

COVER

Syntax

```
COVER(#timeUnit,#demand,#storage,[#last/avg])
```

Description

The COVER function, available in the Data Editor only, determines the amount of time in which your inventory is depleted. To do this, it considers the following information:

- The amount of time, in days, in each period
- The demand data
- The inventory data
- Whether the coverage should look forward or backward

Parameters

Parameter	Meaning
#timeUnit	Enter the time unit; the default is days.
#demand	Enter the name of the field that contains the demand information.
#storage	Enter the name of the field that contains the storage information.
#last/avg	This parameter determines how the Data Editor handles the last period. Forward coverage depends on your knowing future demand; but in the last period, no demand for subsequent periods. exists. The #last option tells the Data Editor to assume that the demand in the periods after the last period is identical to the demand in the last period. The #avg option tells the Data Editor to assume that the demand for periods after the last period is an average of all period demand. This parameter is optional; the default is #last.

Example

For example, suppose you want to calculate the number of days in which your inventory is depleted, and you want to display the result in a new Data Editor field called CoverData. In the Formula field of the Expression Editor for the CoverData field, enter:

```
COVER(#days,#Demand,#Storage,#last)
```

The parameters of this example have the following meanings:

- # *days* is the number of days in each period.
- *#Demand* contains the demand data.
- *#storage* contains the inventory data.
- *#last* means that the demand after the last period is identical to the demand in the last period.

Using the #timeUnit Parameter

The #timeUnit parameter has different meanings in different situations. How the Data Editor interprets the #timeUnit value depends on the situation. Three possible situations exist.

Option 1

Use this option if you do not need a period durations file because your period boundaries are real calendar dates such as February 1, 2005.

This option subtracts the date of the previous period from the date of the current period to determine the length of the period.

For example, if the current period is February 1, 2005, and the previous period is January 1, 2005, 31 days exist in the period. The result of this calculation is dependent on the time unit that you specify in the #timeUnit parameter.

Time Unit	Meaning	Result in Above Example
Days	The number of days between the two dates.	31 days

Weeks	The number of weeks between the two dates. This number is derived by dividing the number of days by 7.	4.4 weeks
Months	The number of months between the dates. This number is the exact number of months if the periods begin on the same day of the month. If the periods do not begin on the same day of the month, the Data Editor assumes that there are 30 days in a month.	1

Option 2

Use this option if you need to have a period durations file because your periods are named by strings, not by real calendar dates such as Period Two instead of February 1, 2005.

This option looks at the period durations file for your definition of the string that makes up the period name. For example, if the period is called Period Two, the Data Editor looks for your definition of Period Two in the period durations file.

If the string is not defined in the period durations file, the Data Editor uses the following defaults:

Time Unit	Meaning
Days	20
Weeks	4
Month	1

Option 2

Use this option if you need to have a period durations file because your periods are named by strings, not by real calendar dates such as Period Two instead of February 1, 2005.

This option looks at the period durations file for your definition of the string that makes up the period name. For example, if the period is called Period Two, the Data Editor looks for your definition of Period Two in the period durations file.

If the string is not defined in the period durations file, the Data Editor uses the following defaults:

Time Unit	Meaning
Days	20
Weeks	4
Month	1

Option 3

Use this option if you need to have a period durations file because your period boundaries are real calendar date-such as February 1, 2000-but you do not want to use them. Instead you want to define your own time unit. The time unit defined in the period durations file is used instead of the real time unit.

This option is useful when you want to define the number of work days in a month instead of the number of days. For example, 31 days are in January 2005, but 21 daya are work days. If you want

the Data Editor to use the number of work days instead of the number of days (as it did in Option 1), you need to use the period durations file. In it, you define the number of days in January 2005 as 21.

If the Data Editor cannot find the period durations file, it uses the calculation defined in Option 2.

If you want to use a period durations file, you first need to create it.

See Also

Cover

Depletion

Inverse Cover

Inverse Depletion

DATETOUMTSEC

Syntax

```
DATETOUMTSEC( date ): integer
```

Description

The DATETOUMTSEC function takes a localized date, converts it to UMT (Universal Mean Time also, called Greenwich Mean Time), and returns the number of seconds elapsed since January 1, 1970 00:00:00 UMT. This function is the inverse of the ABSDATE function.

See Also

ABSDATE

ADVANCEDAYS

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

DAY

Syntax

```
DAY( date )
```

Description

The DAY function, available only in Advanced Planning Agent, returns an integer from 1 to 31. This function can be useful if you want to compare two dates to see if they are on the same day of the month, or if you want to do date arithmetic, such as adding one day to a date. If the value is null, the function returns null.

Example

For example, if the field Period contains the dates January 4, 2005; January 13, 2005; and January 4, 2005; and you want to select rows of data for only the fourth day of the month, enter the following as part of the selection criteria:

```
DAY($1.Period)=4
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

DAYDUR

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

DAYDUR

Syntax

```
DAYDUR( numDays )
```

Description

The DAYDUR function, available only in Advanced Planning Agent, returns the number of seconds in the specified number of days. If no number of days is specified, the default of one day is used. Make sure that you account for daylight saving time changes. This function is useful for date arithmetic. If the value is null, the function returns null.

Example

If you want to add seven days to a date from a field called Date, enter the following output expression:

```
$1.Date + DAYDUR(7)
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

DAY

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

Depletion

Syntax

```
Depletion(# depletor, depletorOffset,# depletee,# timeUnit,
#f/b,#last/avg)
```

Description

The Depletion function is available in the Data Editor only. This function plans coverage. Coverage compares inventory levels to projected demand and determines your ability to meet future demand based on the inventory levels.

The Depletion function determines your coverage requirements. It considers the demand that you have in each period, and determines how long the inventory will meet the demand. To do this, it considers the following information:

- Current and projected inventory levels
- Projected demand
- Unit of time in which the coverage is reported
- Whether the coverage should look forward or backward:

Parameters

Parameter	Meaning
#depletor	This field contains the product demand on which the coverage is based.
depletorOffset	The starting period offset that determines the period, relative to the current period, from which the demand should be calculated. This value must be an integer. For example, if you enter 0, the calculation begins in the current period. If you enter 1, the calculation begins in the next period. Whether it begins 1 period forward or backward depends on the #f/b value.
#depletee	The field that contains the amount of planned inventory or safety stock. The Data Editor considers this field when calculating the amount of available inventory that can cover the demand. If you want the Data Editor to consider how your safety stock value covers demand, you must enter the safety stock field name as the depletee. If you want the Data Editor to consider how your inventory levels compare to your demand, you must enter the storage field name as the depletee.
#timeUnit	Specifies the time units in which the result of the calculation will be measured.You can use days, weeks, or months.
#f/b	Specifies whether the calculation should run forward in time (f) or backward in time (b). This parameter is optional; the default is #f.
#last/avg	Determines how the Data Editor handles the last period. Forward coverage depends on your knowing future demand, but in the last period, no demand exists for subsequent periods. The last option tells the Data Editor to assume that the demand in the periods after the last period is identical to the demand in the last period. The avg option tells the Data Editor to assume that the demand for periods after the last period is an average of all period demand. This parameter is optional; the default is #last.

Example

For example, suppose that you want to calculate the weeks of supply that are available to meet your demand and you want to display the result in a new Data Editor field called WeeksofSupply. You would enter the following in the Formula field of the Expression Editor for the WeeksofSupply field:

```
Depletion(#Demand,1,#EndingInv,#week,#f,#last)
```

The parameters of this example have the following meanings:

Parameter	Meaning
#Demand	Cover is based on the demand in this field.
1	The calculation begins in the next time period.
#EndingInv	This field is compared to the #Demand field to determine the amount of inventory that is available to cover demand.
#f	The depletorOffset begins calculations to project demand forward.
#last	The demand after the last period is identical to the demand in the last period.

EQTOL

Syntax

```
EQTOL ( x,y,tol )
```

Description

The EQTOL function returns TRUE() if the magnitude of the difference between *x* and *y* is less than or equal to the tolerance value (*tol*). The tolerance level is an arbitrary value that you determine, based on your operations. If a value is null, the function returns null.

The EQTOL function returns true if:

$x-y \leq tol$

ERROR

Syntax

```
ERROR( '#errorMessage' )
```

Description

The ERROR function is available in the Data Editor only.

You define the error message. For example, if you want to have a warning message indicating a violation that Demand is below minimum, you could enter *Demand is below Minimum* to indicate that the result of an expression is below a minimum value, such as in the following expression:

```
IF(Demand<=60,ERROR('#Demand is below minimum'),Demand)
```

See Also

GETOL

LETOL

EXP

Syntax

```
EXP( x )
```

Description

The EXP function returns e to the power of x. If the value is null, the function returns null.

FALSE

Syntax

```
FALSE()
```

Description

The FALSE function returns the value 0.

FINDNEXTDAY

Syntax

```
FINDNEXTDAY( start_date,weekDayNum,TRUE ()| FALSE ())
```

Description

The FINDNEXTDAY function, available in Advanced Planning Agent only, returns the date for the next occurrence of the weekday specified by the weekDayNum parameter and following the start_date. If a value is null, the function returns null.

The *start_date* is the date from which the function starts searching for the specified weekday. The *weekDayNum* is an integer from 1 to 7, with 1 representing Sunday, 2 representing Monday, and so on.

TRUE() or FALSE() is an optional parameter that you can use to determine what date is returned. If *weekDayNum* is the same as the day of the start_date, specify TRUE() if you want the function to return the *start_date* or FALSE() if you want the function to return the date seven days after the *start_date* . For example, if the *start_date* is a Tuesday and your *weekDayNum* is 3 (which represents Tuesday), then TRUE() returns your start_date, and FALSE() returns the date that is seven days after the start_date. A value of null returns null.

Examples

`FINDNEXTDAY(GETDATE("2005/04/22"),2)` returns April 28, 2005, which is the next Monday after April 22, 2005.

`FINDNEXTDAY(GETDATE("2005/04/21"),2)` returns April 28, 2005, which is the next Monday after April 21 (also a Monday). The function uses the default parameter FALSE().

`FINDNEXTDAY(GETDATE("2005/04/21"),2,TRUE())` returns April 21, 2005.

See Also

AND,OR, and XOR

BETWEEN

IF

TRUE

GAMMA

Description

The GAMMA function is a continuous generalization of the factorial function. If the value is null, the function returns null.

Syntax

```
GAMMA(x)
```

where x is a number.

For positive integers:

GAMMA(x) = (x-1)!

Note

GAMMA is not defined for negative integers.

Example

For example, the function GAMMA(5) is calculated as 4 x 3 x 2 x 1 and returns 24.

The GAMMA function is not available if you are running ADVANCED PLANNING AGENT on Windows NT or Windows 2000.

GETDATE

Syntax

```
GETDATE( str,format )
```

Description

The GETDATE function, available in Advanced Planning Agent only, returns a date that is defined as a string in a specific format. You can specify an optional parameter for the date format. The date and format must be within double quotes. This function is useful if you have a column that is a string data

type with dates, and you want to use the dates in another table. If a value is null, the function returns null.

Example

```
GETDATE("05/12/23 21:22:23","%y/m/%d %H:%M:%S")
```

The GETDATE function returns a NULL value if the year is less than 1902 and greater than 2038. Otherwise, it returns a numeric value representing the date.

Examples of formats:

Format of String	Example	UNIX Format
Year	2005	%Y
year/month	2005/11	%Y/%m
year/month/day	2005/11/23	%Y/%m/%d
year/month/day hour (hour by 24-hour clock)	2005/11/23 14	%Y/%m/%d %H
year/month/day hour/min	2005/11/23 14:53	%Y/%m/%d%H:%M
year/month/day hour/min/sec	2005/11/23 14:53:35	% Y/%m/%d%H:%M:%S

Note

A space must separate the day from the hour. A colon (:) should be used before the minutes or seconds, not a slash.

Example

For example, if Column A of a table contains dates defined as strings in one of the preceding formats, and you want to use those dates in another table, you can use the following output expression:

```
GETDATE($1.ColumnA)
```

The GETDATE *function* supports the *%V* format specifier.

GETDAYSINMONTH

Syntax

```
GETDAYSINMONTH ( date )
```

Description

The GETDAYSINMONTH function, available in Advanced Planning Agent only, returns the number of days that occur in the month of a specified date. If the value is null, the function returns null. This function is useful if you need to calculate daily rates and your data is by month.

Example

If you want the number of days in February 2005, you can use the following expression:

```
GETDAYSINMONTH(MAKEDATE(2005,2))
```

The expression returns 28.

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

GETENV

Syntax

```
GETENV( string )
```

Description

The GETENV function, available only in Advanced Planning Agent, returns the character value of an environment string.

Example

The following expression uses the date of the NMX_CURTIME variable if it is set. Otherwise, it uses the current date:

```
IF (ISNULL(GETENV(NMX_CURTIME)), GETDATE(GETENV(NMX_CURTIME)), NOW())
```

GETOL

Syntax

```
GETOL( x ,y,tol )
```

Description

The GETOL function (within tolerance *x* is greater than or equal to *y*) returns TRUE() if the difference between x and y is equal to or greater than the negative of the tolerance value (*tol*). The tolerance level is an arbitrary value that you determine, based on your operations. In Advanced Planning Agent, if a value is null, the function returns null.

The GETOL function returns true if:

$x \geq (y-tol)$

See Also

EQTOL

LETOL

GREEN

Syntax

```
GREEN()
```

Description

The GREEN function is available in the Data Editor only. This function changes the color of a value to green.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

If you want the value of a field to be displayed in green if the inventory level goes below the safety level, enter:

```
IF('Inventory Level'<'Safety Level',GREEN(),BLACK())
```

See Also

BACKGROUND

BLACK

BLUE

ORANGE

PINK

RED

VIOLET

YELLOW

GTTOL

Syntax

```
GTTOL( x,y,tol )
```

Description

The GTTOL function (within tolerance *x* is greater than *y*) returns TRUE() if the difference between *x* and *y* is greater than the negative of the tolerance value (*tol*). The tolerance level is an arbitrary value that you determine based on your operations. In Advanced Planning Agent, if a value is null, the function returns null.

The GTTOL function returns TRUE if:

```
x>(y-tol)
```

HOUR

Syntax

```
HOUR( date )
```

Description

The HOUR function, available only in Advanced Planning Agent, returns a number from 0 and 23 for the hour in a specified date. This function is useful when you want to evaluate a time. If the value is null, the function returns null.

Example

If the field Period contains the dates January 4, 2005 22:00, January 13, 2005 22:00, and January 4, 2005 14:00, and you want to select rows of data for only the time of 10 p.m., enter the following as part of the selection criteria:

```
HOUR($1.Period)=22
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

HOURDUR

Syntax

```
HOURDUR ( numHours )
```

Description

The HOURDUR function, available only in Advanced Planning Agent, returns the number of seconds in the specified number of hours. If no number of hours is specified, the default of one hour is used. This function is useful for date arithmetic. If the value is null, the function returns null.

Example

To add four hours to the date January 23, 2005 14:00, enter:

```
MAKEDATE(2005,01,23) + HOURDUR(4)
```

To return the date one hour in the future, enter:

```
NOW() + HOURDUR()
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

IF

Syntax

```
IF( cond,x,y )
```

Description

The IF function takes one action if a condition is true and takes another action if the condition is false: if *cond* is true, perform *x*; otherwise, perform *y* . You can use the IF function with several other expressions. In Advanced Planning Agent, when the *cond* is null, the result is null.

Example

In the Data Editor, if you want to display a value in green if the difference between the planned and actual values is equal to the tolerance level of 5, or in red otherwise, enter:

```
IF(EQTOL(Planned,Actual,5),GREEN(),RED())
```

In Advanced Planning Agent, if the following expression is used as an output expression for the field Order_status, the field is set to 1 if the Order_qty is greater than zero. Otherwise the field is set to 0:

```
IF($1.Order_qty > 0, 1, 0)
```

See Also

AND,OR, and XOR

BETWEEN

FALSE

TRUE

INDEX

Syntax

```
INDEX( string,pattern,count )
```

Description

The INDEX function searches for the count occurrence of a pattern in a string and if found, it returns the index of the start of the occurrence. Otherwise, -1 is returned. If the string is null, the function returns null.

See Also

ASCIICONVERT

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

INFINITESIMAL

Syntax

```
INFINITESIMAL()
```

Description

The INFINITESIMAL function, available only in Advanced Planning Agent, returns the value 10-30(1X10E-30), which approximates almost zero for use in some calculations.

INFINITY

Syntax

```
INFINITY()
```

Description

The INFINITY function, available only in Advanced Planning Agent, returns the value 1030(1X10E30), which approximates infinity for use in some calculations.

This function is useful when you divide amounts in an expression where the amount is infinity (displayed by an asterisk in the Data Editor). Without using the INFINITY function, a resulting number could be a large number when you want it to be infinity.

Example

Use an output expression similar to the following, in which the field $1.Amt is not divided by 365 if its value is infinity:

```
IF($1.Amt=INFINITY(),$1.Amt,$1.Amt/365)
```

Note

INFINITY is bigger than any number that can be represented by the integer data types. Assigning INFINITY to an integer field will have uncertain results.

INSERT

Syntax

```
INSERT( string,index,insertString )
```

Description

The INSERT function inserts insertString into string at index and returns the new string. If index is greater than the length of string, string is returned unchanged. Index can be a positive or negative value. Positive index values count from the beginning. Negative index values count from the end. Start the index at 0 for positive values and -1 for negative values. If the string is null, the function returns null.

See Also

ASCIICONVERT

INDEX

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

INT

Syntax

```
INT( x )
```

Description

The INT function returns the integer value of *x* only. It does not return any decimal point values. Use the INT function if you are not concerned with the decimal point values of a number. In Advanced Planning Agent, if the value is null, the function returns null.

Example

The expression *INT(9.999)* returns the integer 9.

InvCover

Syntax

```
InvCover( #timeUnit , #demand , #cover ,[ #last/avg ])
```

Description

The InvCover() function is available in the Data Editor only. It calculates the amount of inventory you need to meet your demand over a certain length of time. This function helps you calculate your safety stock levels.

The syntax of the InvCover function is similar to the syntax for the Cover function, and each of the parameters has the same meaning as the Cover function parameters.

Parameters

Parameter	Meaning
#timeUnit	In this parameter, enter the time unit. See Using the #timeUnit Parameter for the Cover function for a complete description of the different interpretations that this parameter can have.
#demand	In this parameter, the syntax of the InvCover function is similar to the syntax for the Cover function, and each of the parameters has the same meaning as the Cover function parameters. If a value is null, the function returns null . Enter the name of the field that contains the demand information.
#cover	In this parameter, enter the name of the field that contains the data for the amount of time in which the inventory will be depleted.

Parameter	Meaning
#last/avg	This parameter determines how the Data Editor handles the last period. Forward coverage depends on your knowing future demand, but in the last period, no demand exists for subsequent periods. The last option tells the Data Editor to assume that the demand in the periods after the last period is identical to the demand in the last period. The avg option tells the Data Editor to assume that the demand for periods after the last period is an average of all period demand. This parameter is optional; the default is #last.

Example

Suppose you want to calculate the safety stock for the inventory Red, and you want to display the result in a new Data Editor field called RedSafety. You would enter the following in the Formula field of the Expression Editor for the RedSafety field:

```
InvCover(#days,#RedDemand,#RedCover,#last)
```

The parameters of this example have the following meanings:

Parameter	Meaning
#days	The number of days in each period.
#RedDemand	Contains the demand data.
#RedCover	Contains information about how many days that it will take to deplete the inventory.
#last	The demand after the last period is identical to the demand in the last period.

See Also

Cover

InverseDepletion

Syntax

```
InverseDepletion( ... )
```

Description

The InverseDepletion function is available in the Data Editor only. This function allows you to edit the value calculated by the depletion function, and then recalculate the safety stock or storage that you must have to meet your demand.

When you use the inverse depletion function, you must assign the result of the calculation to a specific field. The inverse depletion function is useful if you want to conduct what-if scenarios.

The syntax of this function is similar to the depletion function, and each of the parameters has the same meaning as the depletion function parameters.

Example

If you have four weeks of inventory and you want to determine how much safety stock you need if you change to two weeks of supply, you would enter the following depletion and inverse depletion function expressions in the Expression Editor:

```
Depletion(#Demand,1,Safety,#days,#f,#last)

Assign(#Safety,InverseDepletion(#Demand,1, #'Weeks of
Supply',#weeks,#f,#last)
```

ISNULL

Syntax

```
ISNULL( expr )
```

Description

The ISNULL function is only available in Advanced Planning Agent, and returns 1 if the *expr* is null. Otherwise it returns 0.

KeepBlanks

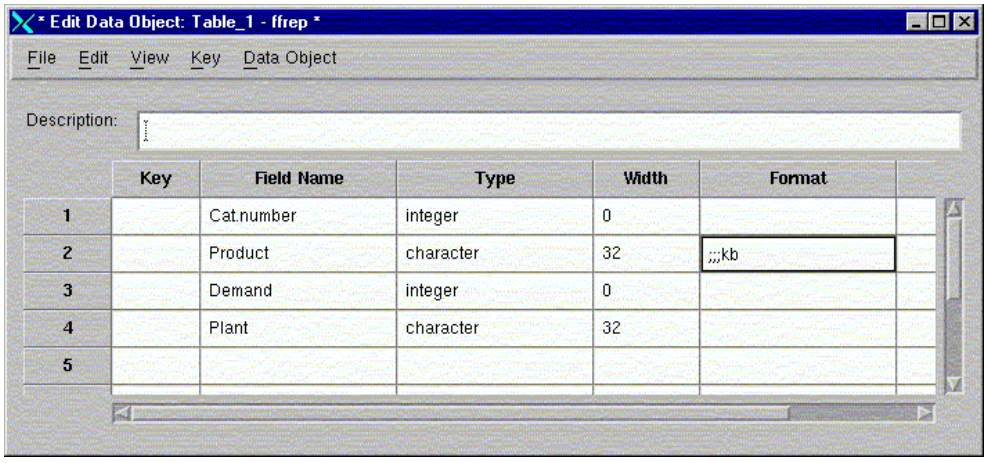
Syntax

```
KeepBlanks( ;;;kb format )
```

Description

The KeepBlanks function is available in the Advanced Planning Agent only. This function allows you to keep trailing blanks. When Advanced Planning Agent reads in data strings, the trailing blanks are stripped by default.

The kb option is placed in the fourth position in the format string. Any other value in the fourth position invokes the default behavior.



KillBlanks

Syntax

```
KillBlanks(string)
```

Description

The KillBlanks function is available in Advanced Planning Agent only. It removes the blanks at the end of a string.

Example

If you have a fixed format length of 10 for an input field but you know that the records that you are outputting require only three spaces for that field, you can use the KillBlanks function to delete the seven empty spaces at the end of each value. If the field is called Product and the value that uses only three spaces is "Red", you can use the following expression:

```
KillBlanks($1.Product)
```

When the function runs, it performs KillBlanks("Red"). The function returns Red without the trailing spaces.

Note

This function may be removed in a future release. Use the TRIMBLANKS, LTRIMBLANKS or RTRIMBLANKS functions instead to ensure future compatibility.

LETOL

Syntax

```
LETOL ( x,y,tol )
```

Description

The LETOL (within tolerance *x* is less than or equal to *y*) function returns TRUE() if the difference between *x* and *y* is less than or equal to the tolerance value (*tol*). The tolerance level is an arbitrary value that you determine based on your operations In Advanced Planning Agent, if a value is null, the function returns null.

The LETOL function returns TRUE if:

$x \leq (y + tol)$

See Also

- *EQTOL*

LN

Syntax

```
LN(x)
```

Description

The LN (natural logarithm) function returns the natural logarithm (base *e*) of an expression (*x*). Use this if you want to perform a statistical analysis that requires logarithmic functions. In Advanced Planning Agent, if the value is null, the function returns null.

LOG

Syntax

```
LOG( x )
```

Description

The LOG (common logarithm) function returns the common logarithm (base 10) of an expression (*x*). Use this if you want to perform a statistical analysis that requires logarithmic functions. In Advanced Planning Agent, if the value is null, the function returns null.

LTRIMBLANKS

Syntax

```
LTRIMBLANKS(string)
```

Description

The LTRIMBLANKS function, available only in Advanced Planning Agent, removes the blanks at the beginning of a string. This function replaces the KillBlanks function, and you should replace any reference to KillBlanks with TRIMBLANKS, LTRIMBLANKS or RTRIMBLANKS.

The LTRIMBLANKS and KillBlanks functions are two interfaces to the same function. Previously, they stripped spaces from the end of string. A second optional parameter has been added to let the user specify the trim character. If not provided, it defaults to a blank space, so it is backwards compatible with no changes to existing flows. A blank space can be a space, tab, carriage return, or another character.

Example

If you have a fixed format length of 10 for an input field but you know that the records that you are outputting require only three spaces for that field, you can use the LTRIMBLANKS function to delete the seven empty spaces at the beginning of each value. If the field is called Product and the value that uses only three spaces is "Red", you can use the following expression:

```
LTRIMBLANKS($1.Product)
```

When the function runs, it performs LTRIMBLANKS("Red") The function returns Red without the leading spaces.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

TRIMBLANKS

LTTOL

Syntax

```
LTTOL ( x,y,tol )
```

Description

The LTTOL function (within tolerance *x* is less than *y*) returns TRUE() if the difference between *x* and *y* is less than the tolerance value (*tol*). The tolerance level is an arbitrary value that you determine, based on your operations. In Advanced Planning Agent, if a value is null, the function returns null.

The LTTOL function returns TRUE if:

```
x < (y + tol)
```

MAKEDATE

Syntax

```
MAKEDATE( year [, month [, day [, hour [, minute [, second ]]]]])
```

Description

The MAKEDATE function, available only in Advanced Planning Agent, returns a date based on the specified numeric values. You must specify the year; the other parameters are optional. If a value is null, the function returns null.

The default value for the month and for the day is 1. The default value for the hour, minute, and second is 0.

If you specify a value that is larger than the normal range for any parameter, the value is carried over to the appropriate parameter. For example, if you specify 24 for the hour parameter, the date rolls over to the next day. Usually, you would specify an hour from 0 to 23.

Example

The following expression returns the date January 13, 2005 at 11:30 a.m.:

```
MAKEDATE(2005,01,13,11,30)
```

The following expression returns the date January 13, 2005 at 12 a.m. (00:00:00):

```
MAKEDATE(2005,01,13)
```

The following expression returns the date that is the first of the current month in the next year:

```
MAKEDATE(NEXTYEAR(NOW()),MONTH(NOW()))
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

MAKEDATEFROMWEEK

Syntax

```
MAKEDATEFROMWEEK(year, weekNum, day, [format option]).
```

Description

The MAKEDATEFROMWEEK function has format options that determine the week number in a year. The week number that is used depends on one of three conventions that are determined by the formatting parameter. The day option stands for day of the week, not day of the month. By default, Sunday is the first day and Saturday is the last day of the week. All days of the specified year that precede the first Sunday of the year are week 0.

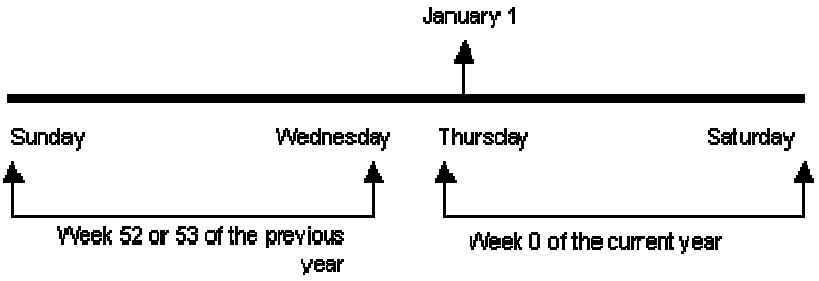
Parameters

Using the %W, %U, and the %V Parameters

You must use the %W, %U, and the %V parameters with double quotes.

Using %U with the MAKEDATEFROMWEEK Function

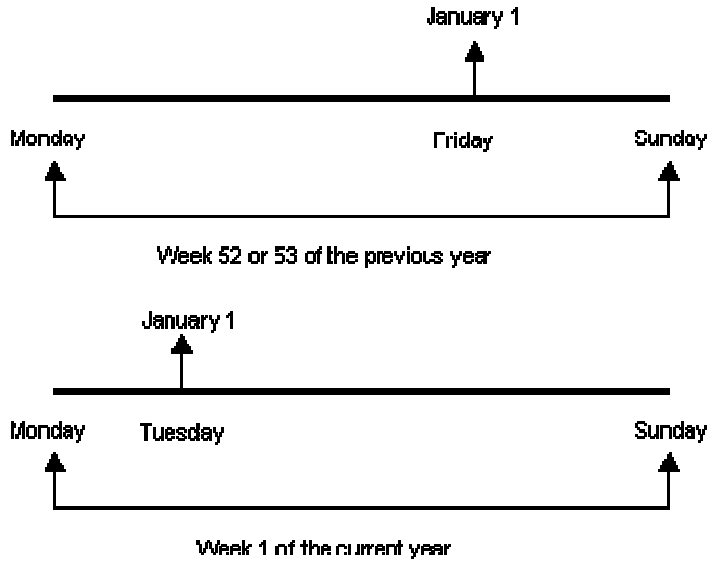
If you use %U, the first day of the week is considered to be Sunday. If January 1st falls on a Sunday, no week 0 exists and the year begins with week 1. Week 0 extends from January 1st to the first Saturday in the year.



Using %U with the MAKEDATEFROMWEEK function

Using %V with the MAKEDATEFROMWEEK Function

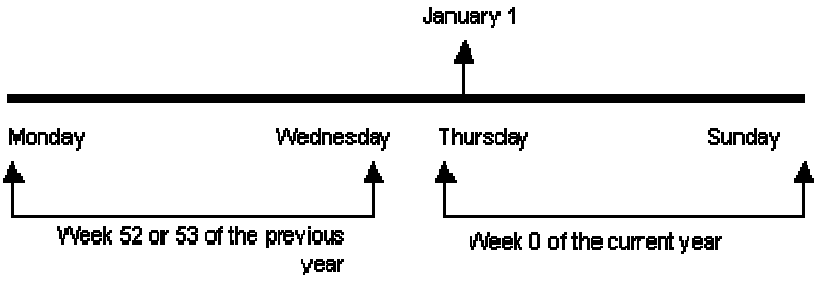
If you use %V, the first day of the week is considered to be Monday. No week 0 exists. If January 1st falls on a Friday, Saturday or Sunday, the entire week is considered to be the last week of the previous year. Week 1 of the current year starts on the following Monday. If January 1st falls on Monday, Tuesday, Wednesday, or Thursday, then the entire week is considered week 1 of the current year.



Using %V with the MAKEDATEFROMWEEK function

Using %W with the MAKEDATEFROMWEEK Function

If you use %W, the first day of the week is considered to be Monday. If January 1st falls on a Monday, there is no week 0 and the year begins with week 1. Week 0 extends from January 1st to the first Sunday in the year.



Using %W with the MAKEDATEFROMWEEK function

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

MAKESYSKEY

Syntax

```
MAKESYSKEY( str )
```

Description

The MAKESYSKEY function, available only in Advanced Planning Agent, creates a system key. The only situation in which you would use this function is when you need to output from a filter data that does not use a system key to a data table that does use a system key. The parameter should be 0, except in rare situations. If the value is null, the function returns null.

The output of the MAKESYSKEY function can only be directed to the System_Key field of the output table (the field created by toggling the system key).

If you require a system key for a field of type syskey that is not the System_Key, use the NEWSYSKEY function.

Example

The following expression creates a system key in an output table:

```
MAKESYSKEY("0")
```

MAX

Syntax

```
MAX( a,b,... )
```

Description

The MAX function returns the maximum value in a list of values. This function is useful if you have a series of calculations and you want to know which expression will return the highest number. If a value is null, it is ignored.

In the Data Editor, the function can only use numeric values such as the following:

```
MAX(1,2,3,4,5)
```

The result of this expression is 5, the maximum number.

Note the following when you use the MAX function in Advanced Planning Agent expressions:

- If the list consists of numbers, the MAX function returns the greatest number.
- If the list consists of character strings, the MAX function sorts the strings alphabetically and returns the greatest string. Lowercase letters are greater than uppercase letters.
- If the list consists of dates, the MAX function sorts the dates chronologically and returns the latest date.
- To use this function in an aggregation filter, specify a single value.

Example

The following expression returns today's date unless *Date* is in the future:

```
MAX(NOW(),$1.Date)
```

MIN

Syntax

```
MIN( a,b,... )
```

Description

The MIN function returns the minimum value in a list of values. This function is useful if you have a series of calculations and you want to know which expression returns the lowest number. If a value is null, it is ignored.

In the Data Editor, the function can only use numeric values, such as the following example:

```
MIN(1,2,3,4,5)
```

The result of this expression is 1, the minimum number.

Note the following when you use the MIN function in Advanced Planning Agent expressions:

- If the list consists of numbers, the MIN function returns the least number.
- If the list consists of character strings, the MIN function sorts the strings alphabetically and returns the least string. Uppercase letters are lesser than lowercase letters.
- If the list consists of dates, the MIN function sorts the dates chronologically and returns the earliest date.
- To use this function in an aggregation filter, specify a single value.

Example

The following expression returns today's date unless Date is in the past:

```
MIN(NOW(),$1.Date)
```

MINUTE

Syntax

```
MINUTE( date )
```

Description

The MINUTE function, available only in Advanced Planning Agent, returns a number from 0 to 59 based on the specified date. This function is useful when you want to evaluate a time. If the value is null, the function returns null.

Example

If the field Period contains the dates January 4, 2005 22:05; January 13, 2005 22:05; and January 4, 2005 14:35; and you want to select rows of data only when the time is five minutes past the hour, enter the following as part of the selection criteria:

```
MINUTE($1.Period)=5
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

MINUTEDUR

Syntax

```
MINUTEDUR( numMinute )
```

Description

The MINUTEDUR function, available only in Advanced Planning Agent, returns the number of seconds in the specified number of minutes. If the number of minutes is not specified, the default of one minute is used. If the value is null, the function returns null. This function is useful for date arithmetic.

Example

The following expression adds forty-five minutes to the date January 23, 2005 14:30:

```
MAKEDATE(2005,1,23,14,30) + MINUTEDUR(45)
```

The following expression returns the date to five minutes in the future:

```
NOW() + MINUTEDIR(5)
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MONTH

Next

NEXTMONTH

NEXT*YEAR*

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

MONTH

Syntax

```
MONTH( date )
```

Description

The MONTH function, available only in Advanced Planning Agent, returns the month number, an integer from 1 to 12, of the specified date. This function is useful if, for example, you want to determine if two dates are in the same month. If the value is null, the function returns null.

Examples

The following expression is true if the date in the Period field is in January:

```
MONTH($1.Period)=1
```

Suppose you want to compare due dates of orders to determine if they are in the same month. This calculation might be necessary if you want to aggregate all of the orders in one month together. To do this, specify the following expression in the Aggregate On window:

```
MONTH($1.Order_date)
```

You might also aggregate by product and specify the following in the Aggregate On window:

```
$1.Product
```

In the Output Expressions window, you would specify the following:

```
SUM($1.Order_quantity)
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

NEG

Syntax

```
NEG( x )
```

Description

The NEG function returns 1 (true) if *x* is negative. Otherwise, it returns 0 (false). In Advanced Planning Agent, if the value is null, the function returns null.

NETOL

Syntax

```
NETOL( x,y,tol )
```

Description

The NETOL is the reverse of the EQTOL function and is only available in the Data Editor. The tolerance level is the value that you determine, based on your business needs.

The function returns TRUE if:

□x-y□>tol

`NETOL(11,10,2)` is FALSE because the difference between 11 and 10 is less than or equal to the tolerance value of 2.

`NETOL(12,10,1)` is TRUE because the difference between 12 and 10 is greater than the tolerance value of 1.

See Also

EQTOL

NEWSYSKEY

Syntax

```
NEWSYSKEY()
```

Description

The NEWSYSKEY function, available only in Advanced Planning Agent, creates a unique system key of a field of type syskey. If the value is null, the function returns null. You can use this function in an output expression to produce unique values for each record of a field in an output data object. The field could be used as a system key field for the output data object.

Next

Syntax

```
Next( #field,outOfRangeValue )
```

Description

The Next function is available only in the Data Editor. It returns the value of the next field, the one after the selected field. If no value in this field exists, it returns the value that you specify as the *outOfRangeValue*.

Parameters

This function has two parameters:

- *#field* means the name of the field to which you want to apply the expression.
- *outOfRangeValue* means the number that is used if no number in the next field exists (this number is optional; the default is 0).

This function affects every column in the Data Editor for the specified field (*#field*). If you want to use the value from only one of these fields, use the relative period function.

Example

Suppose you have a Data Editor view with three columns: January, February, and March. Values for each column in the Blue row are:

	January	February	March

Blue	3	2	4

If we create a field called NextColor that displays the results of the Next function for Blue and has an *outOfRangeValue* of 6, the expression for NextColor is:

```
Next(#Blue,6)
```

The Data Editor looks to the next column and returns its value. No columns exist after March, so the Data Editor returns the *outOfRangeValue* of 6.

	January	February	March
Blue	3	2	4
NextColor	2	4	6

NEXTMONTH

Syntax

```
NEXTMONTH( date ,[ number months ])
```

Description

The NEXTMONTH function, available only in Advanced Planning Agent, returns the date that is the specified number of months in the future (or the past) of the specified date. The returned date is the first of the month at time 00:00:00. If no number of months is specified, the default of one month is used. If a value is null, the function returns null.

This function is useful for finding the beginning of a month in the future or the past.

Specifying the number of months as 0 returns the first of the month in which the date occurs.

Example

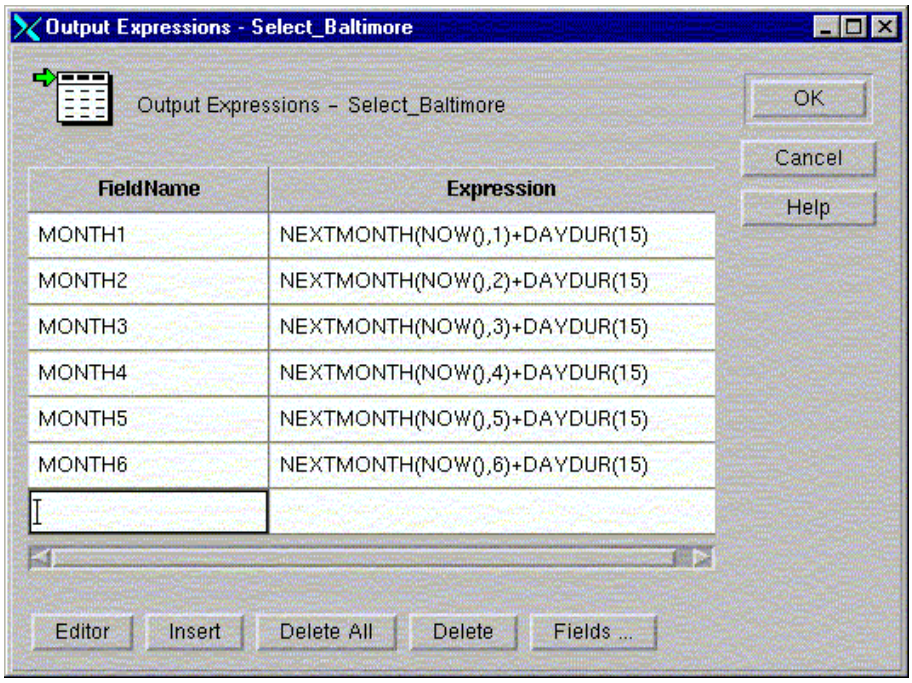
To find the first of the month that is three months after the current month, use the following expression:

```
NEXTMONTH(NOW(),3)
```

To find the first of the month that is two months before the current month, use the following expression:

```
NEXTMONTH(NOW(),-2)
```

You can produce six dates, each on the fifteenth of the month for the next six months, by specifying the following expressions in the Output Expressions window:



NEXTMONTH function

Each date is output to its own field.

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

*HOUR*DUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

NEXTYEAR

Syntax

```
NEXTYEAR( date [, number Years ])
```

Description

The NEXTYEAR function, available only in Advanced Planning Agent, returns the date that is the specified number of years in the future (or the past) of the specified date. The returned date is the first of the year at time 00:00:00. If no number of years is specified, the default of one year is used. If a value is null, the function returns null.

The range that you can use is:

January 1, 1970 to January 19, 2038

This function is useful for finding the beginning of a year in the future or the past. Specifying the number of years as 0 returns the first of the year in which the date occurs. For example, if the value of the Date field is June 13, 2005 at 2 p.m., the following expression returns the date January 1, 2006 at 00:00:00:

```
NEXTYEAR($1.Date)
```

NOT

Syntax

```
NOT( x )
```

Description

The NOT function returns FALSE() if *x* is not 0. If the value is null, the function returns null.

NOW

Syntax

```
NOW()
```

Description

The NOW function is available only in Advanced Planning Agent. It returns the current date and time. This function is useful when you want to specify the current date in other functions.

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

NULL

Syntax

```
NULL( expr )
```

Description

The NULL function, available only in Advanced Planning Agent, returns a null value of the same data type as the expression.

NULLDEF

Syntax

```
NULLDEF( expr1,expr2 )
```

Description

The NULLDEF function, available only in the Advanced Planning Agent, returns *expr2* if *expr1* is null. If *expr1* is not null, the NULLDEF function returns *expr1*. Both expressions must be of the same data type. This function is available only in Advanced Planning Agent.

ORANGE

Syntax

```
ORANGE()
```

Description

The ORANGE function is available in the Data Editor only. It changes the color of a value to orange.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

If you want the value of a field to be displayed in orange if the inventory level goes below the safety level, enter:

```
IF('Inventory Level'<'Safety Level',ORANGE(),BLACK())
```

PERCENT

Syntax

```
PERCENT( x,y )
```

Description

The PERCENT function returns the value of x divided by y multiplied by 100. If x=0 or y=0, the function returns 0. In Advanced Planning Agent, if a value is null, the function returns null.

Example

In a Data Editor expression, if the value of Orders_filled is 10 and the value of Number_orders is 50, the following expression returns 20, indicating that 20% of the orders have been filled:

```
PERCENT(Orders_filled,Number_orders)
```

See Also

BACKGROUND

BLACK

BLUE

GREEN

PINK

RED

VIOLET

YELLOW

OVERUNDER

Syntax

```
OVERUNDER( lb,x,ub )
```

Description

The OVERUNDER function returns FALSE() if *x* is between the lower boundary (*lb*) and the upper boundary (*ub*). If *x* is less than the lower boundary, the Data Editor calculates *lb* - *x* . If *x* is greater than the upper boundary the Data Editor calculates *ub* - *x*. In Advanced Planning Agent, if a value is null, the function returns null.

This function is the reverse of the BETWEEN function. You can use it to ensure that a value is not between two values that you specify.

PARSESTRING

Syntax

```
PARSESTRING( string , delimiter , part )
```

Description

The PARSESTRING function, available only in Advanced Planning Agent, parses a string and returns a substring. The delimiter can be any character, such as a letter, number, or space. If a value is null, the function returns null.

The string is divided at the delimiter into parts (a series of substrings). The part identifies which substring to return.

Example

Suppose you have the following string in the variable string:

The-sky-is-blue-and-the-grass-is-green.

Using the hyphen as the delimiter, you can parse the string and return the fourth substring by entering the following expression:

```
PARSESTRING($1.String,"-",4)
```

The substring blue is returned.

Suppose you have the following string in the variable string:

The sky is blue and the grass is green

Using the letter a as the delimiter, you can parse the string and return the third substring by entering the following expression:

```
PARSESTRING($1.String,"a",3)
```

The substring *ss is green* is returned.

PI

Syntax

```
PI()
```

Description

The PI function returns the value 3.141592. Use this function if you are performing a calculation that requires π.

Example

In an Advanced Planning Agent expression, to calculate the volume of a cylinder (π. *r2.h*), use the following output expression:

```
PI()*POWER($1.radius,2)*$1.height
```

PINK

Syntax

```
PINK()
```

Description

The PINK function is available in the Data Editor only. It changes the color of a value to pink.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

If you want the value of a field to be displayed in pink if the inventory level goes below the safety level, enter:

```
IF('Inventory Level'<'Safety Level',PINK(),BLACK())
```

See Also

BACKGROUND

BLACK

BLUE

GREEN

ORANGE

PINK

RED

VIOLET

YELLOW

POS

Syntax

```
POS( x )
```

Description

The POS function returns TRUE() if *x* is greater than 0. In Advanced Planning Agent, if the value is null, the function returns null.

POWER

Syntax

```
POWER( x,y )
```

Description

The POWER function returns *x* to the power of *y* . This function is similar to the EXP(*x*) function, but it allows you to select the base and the exponent. In Advanced Planning Agent, if a value is null, the function returns null. POWER(0,0) also returns null.

Example

If you want to find the volume of a cube whose length, width, and height are 5, enter:

```
POWER(5,3)
```

This expression results in 125.

PREV

Syntax

```
PREV( #field , outOfRangeValue )
```

Description

The PREV function is available in the Data Editor only. It returns the value of the previous field, the one before the selected field. If no value exists in this field, it returns the value that you specify in the *outOfRangeValue* .

This function has two parameters:

- # *field* is the name of the field to which you want to apply the expression.
- *outOfRangeValue* is a number that is used if no number exists in the previous field (this number is optional; the default is 0).

This function affects every column in the Data Editor for the specified field (*#field*). If you want to use the value from only one of these fields, use the relative period function.

Example

Suppose you have a Data Editor view with three columns—January, February, and March—and the values for the Blue row are as follows:

	January	February	March
Blue	3	2	4

If you create a new field called PrevColor that displays the results of the Prev function for Blue and has an *outOfRangeValue* of 5, the expression for PrevColor is the following:

```
Previous(#Blue, 5)
```

The Data Editor looks to the previous column and returns its value. No columns exist before January, so the Data Editor returns the *outofRangeValue* of 5.

	January	February	March
Blue	3	2	4
PrevColor	5	3	2

PRODUCT

Syntax

```
PRODUCT( a , b , ... )
```

Description

The PRODUCT function multiplies the list of numbers in parentheses. In Advanced Planning Agent, if a value is null, it is ignored.

Example

If you enter the following expression:

```
PRODUCT(1,2,3,4,5)
```

This expression results in 120.

For another example, suppose you have the production cost per unit of Yellow listed in the Yellow Cost field, and you have the number of units of Yellow that you produced listed in the Yellow Produced field. You can multiply the values of the two fields to arrive at the cost of producing Yellow with the following expression:

```
PRODUCT('Yellow Cost','Yellow Produced')
```

In Advanced Planning Agent, the PRODUCT function returns the product of values in an aggregated field. You can use this function only in an aggregation filter.

For example, the following expression returns the product of the values in the aggregated Value field.

```
PRODUCT($1.Value)
```

R_AVG

Syntax

```
R_AVG( numeric_expression , range )
```

Description

The R_AVG function is available only in Advanced Planning Agent. It is used in an aggregation filter to calculate the rolling average. The range specifies the number of records that are averaged, including the current record. If you specify a range of 0, all the records up to the current record are used in the calculation of the rolling average. If a value is null, it is treated as 0.

Example

The following expression calculates the rolling average using the values in the Demand field. The current record and the previous record are averaged:

```
R_AVG($1.Demand,2)
```

See Also

R_COVER

R_MAX

R_MIN

R_PREV

R_PRODUCT

R_SUM

R_COVER

Syntax

```
R_COVER ( x , y [ ,flag ][, period ][, units ])
```

Description

The R_COVER function is used only in Advanced Planning Agent in an aggregation filter to compute the periods of coverage for an inventory level. The x is the demand level for the period; y is the

inventory level for the period. The flag, which is optional, is b for beginning inventory (the default) or e for ending inventory. The period is the size of the period, units is the time unit: h for hours, d for days, w for weeks, or m for months. If you specify m, a 28-day month is assumed. If a value is null, it is treated as 0.

The units parameter of the R_COVER function *n* is optional and independent of the periods parameter.

The arguments *flag* and *units* for this function must be inside double quotes.

Example

Consider the following expression:

```
R_COVER(Demand,Inventory,"b",1,"d")/DAYDUR()
```

If you do not specify a period size and units value, the R_COVER function returns the number of periods covered. If you do specify a period size and units value, it returns the actual time of coverage in seconds. You can convert the output to days by dividing by the DAYDUR function.

For this function to compute the days cover, you must ensure that the inventory and demand pairs are provided to the function in the required reverse order. You can do this by setting the sort order button. Demand beyond the horizon is assumed to be the same as the last demand value.

For example, consider the following data:

Product	Period	Demand	Inventory	Periods Covered (Beginning Inventories)	Periods Covered (Ending Inventories)
A	P1	10	40	2.0	1.4
A	P2	30	50	1.8	2.25
A	P3	25	15	0.6	0.75
A	P4	20	30	1.5	1.5
B	P1	20	20	1.0	1.0

This data is processed by the R_COVER function in reverse order (that is, P4 before P3, and so on). The calculations for ending inventory use the demand from the previous row, in the sorted order. The calculations for product A, period P3, are as follows:

- For beginning inventories:
  Coverage = Inventory / Demand = 15/25 = 0.6
- For ending inventories: Coverage = Inventory / Demand = 15/20 = 0.75

The standard use of the R_COVER function is to compute the demand that an inventory level can cover in some unit, often days. For example, consider the following data:

Date	Demand	Inventory	Product
day1	7	15	A

day2	10	12	A
day3	5	10	A
day4	9	8	A
day5	6	12	A
day1	10	25	B
day2	11	15	B
day3	7	7	B
day4	9	19	B
day5	0	10	B

Suppose the R_COVER function is used as follows:

```
R_COVER(Demand,Inventory,b,1,d)/DAYDUR()
```

Because R_COVER returns a number of seconds, dividing by DAYDUR() results in the number of days of cover. Consider day1 to day3 for product A. The inventory for day1, 15, can cover all of the demand for day1 plus 0.8 of the demand on day2. Therefore, the days cover for the inventory level beginning on day1 is 1.8.

If the third parameter in the R_COVER function is e instead of b, then the inventory represents the inventory level at the end of day1. This inventory can then cover the demand for day2 and day3. The days cover for the inventory at the end of day1 would be 2 days.

Some care is needed to use the R_COVER function with the aggregation filter. The aggregation filter sorts the data by the sort fields. Continuing with the same example data, suppose the filter is filled in as follows:

```
Reset Key $1.Product +
Roll Key $1.Date -
```

The filter would sort the data as follows:

Product	Date	Demand	Inventory	Cover
A	day5	6	12	2
A	day4	9	8	0.889
A	day3	5	10	1.556
A	day2	10	12	1.4
A	day1	7	15	1.8
B	day5	0	10	Infinity
B	day4	9	19	Infinity

B	day3	7	7	1
B	day2	11	15	1.571
B	day1	10	25	2.571

The aggregation filter evaluates the functions on the data in the order in which it is sorted. The R_COVER function remembers the demand that it has previously read.

If the Date field is not sorted in reverse order, the R_COVER function needs the value of the Demand field for day2, which it has not read yet. However, if it is sorted in reverse order, then day2 has been read before the cover value for day1 is evaluated.

See Also

R_AVG

R_MAX

R_MIN

R_PREV

R_PRODUCT

R_SUM

R_MAX

Syntax

```
R_MAX( expression , range )
```

Description

The R_MAX function is available only in Advanced Planning Agent. It is used in an aggregation filter to calculate the rolling maximum. The range specifies the number of records that are considered, including the current record. If you specify a range of 0, all the records up to the current record are used to determine the rolling maximum. If a value is null, it is treated as 0.

The expression can be a date, string, or number. The R_MAX function returns the maximum value as follows:

- If the expression consists of numbers, the R_MAX function returns the greatest number.
- If the expression consists of character strings, the R_MAX function sorts the strings alphabetically and returns the greatest string. Lowercase letters are greater than uppercase letters.
- If the expression consists of dates, the R_MAX function sorts the dates chronologically and returns the latest date.

See Also

R_AVG

R_COVER

R_MIN

R_PREV

R_PRODUCT

R_SUM

R_MIN

Syntax

```
R_MIN( expression , range )
```

Description

The R_MIN function is available only in Advanced Planning Agent. It is used in an aggregation filter to calculate the rolling minimum. The range specifies the number of records that are considered, including the current record. If you specify a range of 0, all the records up to the current record are used to determine the rolling minimum. If a value is null, it is treated as 0.

The expression can be a date, string, or number. The R_MIN function returns the minimum value as follows:

- If the expression consists of numbers, the R_MIN function returns the least number.
- If the expression consists of character strings, the R_MIN function sorts the strings alphabetically and returns the least string. Uppercase letters are less than lowercase letters.
- If the expression consists of dates, the R_MIN function sorts the dates chronologically and returns the earliest date.

See Also

R_AVG

R_COVER

R_MAX

R_PREV

R_PRODUCT

R_SUM

R _PREV

Syntax

```
R_PREV( expression , default , offset )
```

Description

The R_PREVfunction is used only in Advanced Planning Agent in an aggregation filter to specify that you want to use a previous value. The expression indicates the field that is used. The default can

be an expression that indicates the value used if no previous value exists (according to the offset). The offset indicates the position of the record, back from the current record, that refers back to the function. If a value is null, it is treated as 0.

Example

The expression *R_PREV($1.Demand,0,4)* specifies that you want to use the value from the Demand field that is four records back. If no record exists that far back, a default of 0 is used.

This example shows how two different R_PREV work, using the data in the following table:

Row	Value
1	10
2	15
3	2.5
4	3.1
5	7
6	14

The following tables show the results from two different output expressions.

The first output expression, *R_PREV($1.Value,2,2)* , means that the function uses the data from the row in the Value field that is two rows back. If no such row exists, the default value of 2 is used.

The second output expression, *R_PREV($1.Value,4,1)* , means that the function uses the data from the row in the Value field that is one row back. If no such row exists, the default value of 4 is used.

Output Expression: R_PREV($1.Value,2,2)	
Row	**Output**
1	2
2	2
3	10
4	15
5	2.5
6	3.1

Output Expression: R_PREV($1.Value,4,1)	
Row	**Output**
1	4
2	10
3	15
4	2.5
5	3.1
6	7

See Also

R_AVG

R_COVER

R_MAX

R_MIN

R_PRODUCT

R_SUM

R_PRODUCT

Syntax

```
R_PRODUCT( numeric_expression , range )
```

Description

The R_PRODUCT function is used only in Advanced Planning Agent in an aggregation filter to calculate the rolling product. The range specifies the number of records that are multiplied, including the current record. If you specify a range of 0, all the records up to the current record are used in the calculation of the rolling product. If a value is null, it is treated as 0.

See Also

R_AVG

R_COVER

R_MAX

R_MIN

R_PREV

R_SUM

R_SUM

Syntax

```
R_SUM(SUM( numeric_expression) , range )
```

Description

The R_SUM function is used only in Advanced Planning Agent in an aggregation filter to calculate the rolling sum. The range specifies the number of records that are added, including the current record. If you specify a range of 0, all the records up to the current record are used in the calculation of the rolling sum. If a value is null, it is treated as 0.

See Also

R_AVG

R_COVER

R_MAX

R_MIN

R_PREV

R_PRODUCT

RAND

Syntax

```
RAND( x,y )
```

Description

The RAND function returns a random number between *x* and *y-1* . If a value is null, the function returns null.

Use this function if you want to model your capacity based on your demand, but you do not know your demand. The random numbers could be used to project demand.

Example

The following expression returns a random number from 0 to 99:

```
RAND(0,100)
```

If *x* is greater than or equal to *y* , the function returns *x* . For example, the following returns the number 99:

```
RAND (99,45)
```

RED

Syntax

```
RED()
```

Description

The RED function is available in the Data Editor only. It changes the color of a value to red.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

If you want the value of a field to be displayed in red if the inventory level goes below the safety level, you enter:

```
IF('Inventory Level'<'Safety Level',RED(),BLACK())
```

REGEXPR

Syntax

```
REGEXPR( string , pattern )
```

Description

The REGEXPR function is available only in Advanced Planning Agent. It returns a substring by matching a regular expression pattern to a string. If a value is null, the function returns null.

See Also

BACKGROUND

BLACK

BLUE

GREEN

ORANGE

PINK

RED

VIOLET

YELLOW

RelOffset

Syntax

```
RelOffset( ... )( relative offset )
```

Description

The RelOffset function is available in the Data Editor only. It returns a value from a relative position.

Parameters

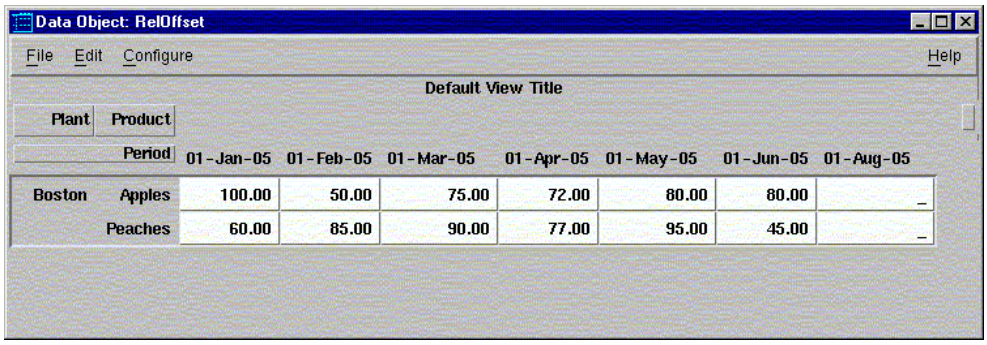
This function has three parameters:

- *#field* is the field name that you select.
- *offset* is the number of fields relative to the field that you specify in *#field* . A positive number looks to the right from this field while a negative number looks to the left from this field.
- *outOfRangeValue* is the number that is used if no column exists or a null value is found by the *offset* value. The default for *outOfRangeValue* is 0.

This function affects every column in the Data Editor for the specified field (*#field*). If you want to use the value from only one of these fields, use the relative period function.

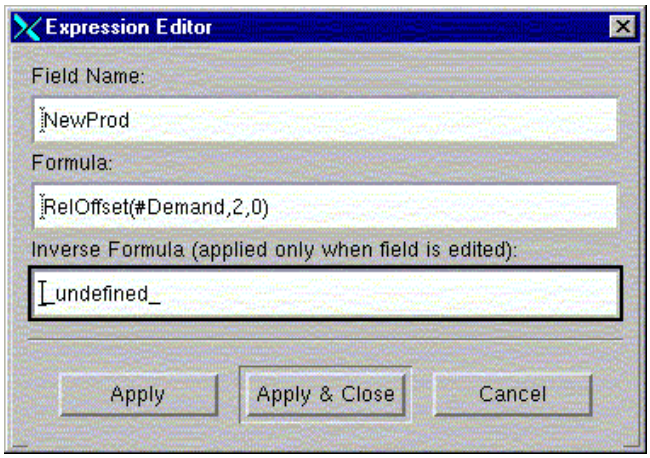
Example

Consider the Data Object table as displayed in the following Data Editor view:



Data object table

Create a field called NewProd that displays the results of the RelOffset function for Demand that is offset by 2 and has an *outofRangeValu e* of 0. As shown in the following Expression Editor, in the Formula field enter: *RelOffset(#Demand,2,0). In the Field Name field, enter NewProd.*



RelOffset expression

The RelOffset function processes the data as follows:

- The entry in the field for the NewProd row and the 01-Jan-05 column is populated with the value from the field that is 2 columns to the right in the Demand row, which is the 01-Mar-05 column with a value of 75.
- The entry in the field for the NewProd row and the 01-Feb-05 column is populated with the value from the field that is 2 columns to the right in the Demand row, which is the 01-Apr-05 column with a value of 72.
- The entry in the field for the NewProd row and the 01-Mar-05 column is populated with the value from the field that is 2 columns to the right in the Demand row, which is the 01-May-05 column with a value of 80.
- The entry in the field for the NewProd row and the 01-Apr-05 column is populated with the value from the field that is 2 columns to the right in the Demand row, which is the 01-Jun-05 column with a value of 80.

- The entry in the field for the NewProd row and the 01-May-05 column is populated with the value from the field that is 2 columns to the right in the Demand row, which is the 01-Aug-05 column with a null value.
- The entry in the field for the NewProd row and the 01-Jun-05 column is populated with a null value because 2 columns do not exist to the right of the 01-Aug-05 column in the Demand row.

Using the Value from a Single Relative Period

The Next, Previous, and RelOffset functions give you values from relative periods for every column in the Data Editor. However, you may need to use the value from a single relative period, and you do not need all of the information that the Next, Previous, and RelOffset functions give you.

In this case, you can use the relative period parameters. Relative period parameters locate one period and use its value in your expression. For example, you might want to use the value in the Min field that is two periods ahead of the current period.

If you specify a period that is after the last period, the data in the last period will be repeated. If you specify a period that is before the first period, the data in the first period will be repeated.

The following table shows how to use relative periods:

Period	Syntax	Explanation
Current	fieldname	This is the usual method. The expression works with the data in the current period.
Previous	fieldname:-n	The number of periods previous, relative to the current period. For example, Min:-3 means to use the value from the Min field that is three periods before the current period.
Future	fieldname:n	The number of periods in the future, relative to the current period. For example, Min:3 means to use the value from the Min field that is three periods after the current period.
Specific	fieldname:#n	The specific periods to use, regardless of the current period. For example, Min:#3 means to use the value from the Min field in the third period (the first period in your model is numbered 1).

REMAINDER

Syntax

```
REMAINDER( x , y )
```

Description

The REMAINDER function, available only in Advanced Planning Agent, returns the remainder of *xy*. Both *x* and *y* must be integers. If a value is null, the function returns null.

Example

The following expression returns 2:

```
REMAINDER(20,6)
```

The Advanced Planning Agent treats numbers greater than or equal to 1E30 as infinity, and numbers less than or equal to 1E-30 as zero. If the first number of the REMAINDER function is greater than or equal to 1E30 or the divisor is less than or equal to 1E-30, the result is null.

REMOVE

Syntax

```
REMOVE( string , index , length )
```

Description

The REMOVE function removes length characters from string, starting at index and returns the new string. If index is greater than the length of string, returns string unchanged. Length and index can be positive or negative values. Positive index and length values count from the beginning. Negative index and length values count from the end. Start the index at 0 for positive values and -1 for negative values. If the string is null, the function returns null.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

REPLACEINDEX

Syntax

```
REPLACEINDEX( string , index , length , replacestring )
```

Description

The REPLACEINDEX function replaces length characters in string, starting at index with replacestring and returns the new string. If index is greater than the length of string, string is returned unchanged. Length and index can be positive or negative values. Positive index and length values count from the beginning. Negative index and length values count from the end. Start the index at 0 for positive values and -1 for negative values. If the string is null, the function returns null.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

REPLACEPATTERN

Syntax

```
REPLACEPATTERN( string , pattern , replaceString )
```

Description

The REPLACEPATTERN function searches for the first occurrence of *pattern* in the *string*. If a *pattern* is found, this function replaces the occurrence with replaceString and returns the new string. Otherwise, returns string unchanged. If the string is null, the function returns null.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

ROUND

Syntax

```
ROUND( x [, num decimal places ])
```

Description

The ROUND function rounds a number to the specified number of decimal places. The default number of decimal places is 1. If a value is null, the function returns null.

Examples

The following expression returns 123.769.

```
ROUND(123.7685,3)
```

The following Advanced Planning Agent expression round the values in the Total_Cost field to two decimal places:

```
ROUND($1.Total_Cost,2)
```

RTRIMBLANKS

Syntax

```
RTRIMBLANKS( string )
```

Description

The RTRIMBLANKS function, available only in Advanced Planning Agent, removes the blanks at the end of a string. This function replaces the KillBlanks function, and you should replace any reference to KillBlanks with RTRIMBLANKS.

The RTRIMBLANKS and KillBlanks functions are two interfaces to the same function. Previously, they stripped spaces from the end of string. A second optional parameter has been added to let the user specify the trim character. If not provided, it defaults to a blank space, so it is backwards compatible with no changes to existing flows. A blank space can be a space, tab, carriage return, or another character.

Example

If you have a fixed format length of 10 for an input field but you know that the records that you are outputting require only three spaces for that field, you can use the RTRIMBLANKS function to delete the seven empty spaces at the end of each value. If the field is called Product and the value that uses only three spaces is "Red ", you can use the following expression:

```
RTRIMBLANKS($1.Product)
```

When the function runs, it performs RTRIMBLANKS("Red ") The function returns Red without the trailing spaces.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

LTRIMBLANKS

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

TRIMBLANKS

SECOND

Syntax

```
SECOND( date )
```

Description

The SECOND function, available only in Advanced Planning Agent, returns a number from 0 to 59, which is the seconds part of the time in the specified date. If a value is null, the function returns null. This function can be useful when you want to compare two times to see if they are the same.

Example

If your data contains the dates January 4, 2005 22:10:45; January 13, 2005 22:10:45; and January 4, 2005 14:35:25; and you want to select only the dates in which the time includes 45 seconds, enter:

```
SECOND($1.Period)=45
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

SECONDDUR

Syntax

```
SECONDDUR(numSeconds)
```

Description

The SECONDDUR function, available only in Advanced Planning Agent, returns the number of seconds specified. If no number of seconds is specified, the default of one second is used. If a value is null, the function returns null.

Example

If you want to add 45 seconds to dates that are output to a data table, enter:

```
$1.Date+SECONDDUR(45)
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

SEQUENCE

Syntax

```
SEQUENCE( a , b , ... )
```

Description

The SEQUENCE(*a,b,...*) function is available in the Data Editor only. It ensures that the values or expressions are performed in the order in which they appear and returns TRUE(). This function is useful if you have multiple entries of the Assign function in an inverse expression.

SIGN

Syntax

```
SIGN(x)
```

Description

The SIGN function returns -1 if *x* is less than 0, returns 0 if *x* is equal to 0, or returns 1 if *x* is greater than 0. In Advanced Planning Agent, if the value is null, the function returns null.

SIN

Syntax

```
SIN(x)
```

Description

The SIN function, available only in Advanced Planning Agent, returns the sine of *x* , where *x* is measured in radians. If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

ATAN2

COS

COSH

SINH

TAN

TANH

SINH

Syntax

```
SINH(x)
```

Description

The SINH function, available only in Advanced Planning Agent, returns the hyperbolic sine of *x* , where *x* is measured in radians. If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

ATAN2

COS

COSH

SIN

TAN

TANH

SQRT

Syntax

```
SQRT(x)
```

Description

The SQRT function returns the positive square root of a number. In Advanced Planning Agent, if the value is null, the function returns null.

Example

The following expression results in 3.87:

```
SQRT(15)
```

STDDEV

Syntax

```
STDDEV(a , b , ...)
```

Description

The STDDEV function is available only in Advanced Planning Agent. It returns the standard deviation of the values in the list. If a value is null, it is ignored.

The STDDEV(*numeric_expression*) function returns the standard deviation of values in an aggregated field. You can use this only in an aggregation filter.

STRINGCAST

Syntax

```
STRINGCAST(date or number [, format string])
```

Description

The STRINGCAST function, available only in Advanced Planning Agent, converts a number or a date to a string. If a value is null, the function returns null.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

STRINGCONCAT

Syntax

```
STRINGCONCAT(a , b , ...)
```

Description

The STRINGCONCAT function, available only in Advanced Planning Agent, concatenates one or more strings. If a value is null, the function returns null.

Example

If you want to create a unique code name string for a product based on the product characteristics that are separately identified in their own fields, enter:

```
STRINGCONCAT($1.Brand,$1.Colour,$1.Volume,$1.Label)
```

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

STRINGLENGTH

Syntax

```
STRINGLENGTH(string)
```

Description

The STRINGLENGTH function, available only in Advanced Planning Agent, returns the number of characters in a string. If a value is null, the function returns null. This is useful information when you want to perform other string manipulation operations.

Example

If the variable Label has the value XL48, the following expression returns 4:

```
STRINGLENGTH($1.Label)
```

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

STRINGTODOUBLE

Syntax

```
STRINGTODOUBLE(string)
```

Description

The STRINGTODOUBLE function, available only in Advanced Planning Agent, converts a numeric string from character to double. If the value is null, the function returns null.

Example

The following expression returns the value 10.345 of data type double:

```
STRINGTODOUBLE("10.345")
```

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

STRINGTOINT

Syntax

```
STRINGTOINT(string)
```

Description

The STRINGTOINT function, available only in Advanced Planning Agent, converts a numeric string from character to integer. If the value is null, the function returns null.

Example

The following expression returns the integer 10:

```
STRINGTOINT("10")
```

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

SUBSTRING

TOLOWER

TOUPPER

SUBSTRING

Syntax

```
SUBSTRING(string , x , y)
```

Description

The SUBSTRING function is available in Advanced Planning Agent only. It returns a substring of length *y* starting at character x (that is, the *x* th character) in the specified string. If a value is null, the function returns null.

Example

If the field Product contains the string *12ozDarkAle,* the following expression returns the string Dark:

```
SUBSTRING($1.Product,5,4)
```

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

TOLOWER

TOUPPER

SUM

Syntax

```
SUM(a , b , ...)
```

Description

The SUM function returns the sum of a list of values or expressions. In Advanced Planning Agent, the SUM(*numeric_expression*) function returns the sum of values in an aggregated field. If a value is null, it is ignored. You can use this only in an aggregation filter.

Example

The following function returns 15:

```
SUM(1,2,3,4,5)
```

The following expression returns the sum of the values in the aggregated Demand field:

```
SUM($1.Demand)
```

TAN

Syntax

```
TAN(x)
```

Description

The TAN function, available only in Advanced Planning Agent, returns the tangent of *x*, where *x* is measured in radians. If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

ATAN2

COS

COSH

SIN

SINH

TANH

TANH

Syntax

```
TANH(x)
```

Description

The TANH function, available only in Advanced Planning Agent, returns the hyperbolic tangent of *x*, where *x* is measured in radians. If the value is null, the function returns null.

See Also

ACOS

ASIN

ATAN

ATAN2

COS

COSH

SIN

SINH

TAN

TOLOWER

Syntax

```
TOLOWER(string)
```

Description

The TOLOWER function converts all alphabetic characters in a string to lower case and returns the new string. If the string is null, the function returns null.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOUPPER

TOUPPER

Syntax

```
TOUPPER(string)
```

Description

The TOUPPER converts all alphabetic characters in string to upper case and returns the new string. If the string is null, the function returns null.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TRIMBLANKS

Syntax

```
TRIMBLANKS(string)
```

Description

The TRIMBLANKS function, available only in Advanced Planning Agent, removes the blanks at the beginning and end of a string. This function replaces the KillBlanks function, and you should replace any reference to KillBlanks with TRIMBLANKS, LTRIMBLANKS or RTRIMBLANKS.

The TRIMBLANKS and KillBlanks functions are two interfaces to the same function. Previously, they stripped spaces from a string. A second optional parameter has been added to let the user specify the trim character. If not provided, it defaults to a blank space, so it is backwards compatible with no changes to existing flows. A blank space can be a space, tab, carriage return, or another character.

Example

If you have a fixed format length of 10 for an input field but you know that the records that you are outputting require only three spaces for that field, you can use the TRIMBLANKS function to delete the seven empty spaces at the start and end of each value. If the field is called Product and the value that uses only three spaces is "Red ", you can use the following expression:

```
TRIMBLANKS($1.Product)
```

When the function runs, it performs TRIMBLANKS("Red") The function returns Red without the leading or trailing spaces.

See Also

ASCIICONVERT

INDEX

INSERT

KeepBlanks

KillBlanks

LTRIMBLANKS

PARSESTRING

REGEXPR

REMOVE

REPLACEINDEX

REPLACEPATTERN

RTRIMBLANKS

STRINGCAST

STRINGCONCAT

STRINGLENGTH

STRINGTODOUBLE

STRINGTOINT

SUBSTRING

TOLOWER

TOUPPER

TRUE

Syntax

```
TRUE()
```

Description

The TRUE function returns the value 1.

See Also

AND, OR, and XOR

BETWEEN

FALSE

IF

VIOLET

Syntax

```
VIOLET()
```

Description

The VIOLET function is available in the Data Editor only. It changes the color of a value to violet.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

If you want the value of a field to be displayed in violet if the inventory level goes below the safety level, you enter the following:

```
IF('Inventory Level'<'Safety Level',VIOLET(),BLACK())
```

See Also

*B*ACKGROUND

BLACK

BLUE

GREEN

ORANGE

PINK

RED

YELLOW

WEEKDAY

Syntax

```
WEEKDAY(date)
```

Description

The WEEKDAY function, available only in Advanced Planning Agent, returns the day of the week as an integer, where 1 is Sunday, 2 is Monday, and so on. If the value is null, the function returns null.

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDUR

WEEKNUMBER

YEAR

YEARDAYNUMBER

WEEKDUR

Syntax

```
WEEKDUR(numWeeks)
```

Description

The WEEKDUR function, available only in Advanced Planning Agent, returns the number of seconds in the specified period of weeks. If the value is null, the function returns null.

Example

To add four weeks to the current date, enter:

```
NOW()+WEEKDUR(4)
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKNUMBER

YEAR

YEARDAYNUMBER

WEEKNUMBER

Syntax

```
WEEKNUMBER( date )
```

Description

The WEEKNUMBER function, available only in Advanced Planning Agent, returns a week number from 0 to 51. If the value is null, the function returns null.

This function is useful if you want to compare two dates to see if they occur in the same week. Week 1 starts on the first Sunday of the year. Week 0 is for the days that precede the first Sunday of the year.

Example

To select dates from a field called Date only if they are in week 10, use the following expression:

```
WEEKNUMBER($1.Date)=10
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

YEARDAYNUMBER

YEAR

Syntax

```
YEAR( date )
```

Description

The YEAR function, available only in Advanced Planning Agent, returns the four-digit year component of the specified date. If the value is null, the function returns null.

Two-digit dates are interpreted as follows:

- A number equal to or greater than 38 is in this century (38 means 1938)
- A number less than 38 is in the next century (37 means 2037)

Example

If the data field Period contains the dates January 4, 2005; January 13, 2005; and January 4, 1996; and you want to select dates only for the year 2005, enter:

```
YEAR($1.Period)=2005
```

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

YEAR

YEARDAYNUMBER

YEARDAYNUMBER

Syntax

```
YEARDAYNUMBER( date )
```

Description

The YEARDAYNUMBER(*date*) function, available only in Advanced Planning Agent, returns the day-number, counting January 1 as day number one. If the value is null, the function returns null.

See Also

ABSDATE

ADVANCEDAYS

DATETOUMTSEC

FINDNEXTDAY

GETDATE

GETDAYSINMONTH

HOUR

HOURDUR

MAKEDATE

MAKEDATEFROMWEEK

MIN

MINUTE

MINUTEDUR

MONTH

Next

NEXTMONTH

NEXTYEAR

NOW

SECOND

SECONDDUR

WEEKDAY

WEEKDUR

WEEKNUMBER

YEAR

YELLOW

Syntax

```
YELLOW()
```

Description

The YELLOW function is available in the Data Editor only. It changes the color of a value to yellow.

Color functions are part of the highlighting option. When you use them, you must enter your expression in the Highlight Expression Editor.

Example

To display a field in in yellow if the inventory level goes below the safety level, enter:

```
IF('Inventory Level'<'Safety Level',YELLOW(),BLACK())
```

See Also

BACKGROUND

BLACK

BLUE

GREEN

ORANGE

PINK

RED

VIOLET

ZERO

The ZERO function returns 1 if *x* is equal to 0. Otherwise, it returns 0.

The function syntax is:

```
ZERO(x)
```

Packing Project Files

You use the pack utility to pack and send projects and all its related data from one site to another. When you pack a project, the data is stored in a single directory, which can then be archived to tape or transmitted by modem. The destination site can unpack the data to use the definitions and run the data flows. You can also select to export selected data. You do all these tasks in the pack utility window.

This section discusses how to:

- Start the pack utility.
- Pack a project.
- Unpack a project.
- Export data using the pack utility.
- Import data using the pack utility.

Starting the Pack Utility

This section discusses how to:

- Start the pack utility in Windows.
- Start the pack utility in UNIX.

Starting the Pack Utility in Windows

To start the pack utility in Windows, select Start, Programs, EnterpriseOne Supply Chain Planning, Advanced Planning Agent *x.x.x* , pack utility.

Note

If you start the pack utility from an Xstart window, enter the following command:

```
" C:/Program Files/EnterpriseOne Supply Chain Planning/APAg/vers_ x.x.x /
bin/APAgAdmin x . x.x . exe"
"C:/Program Files/EnterpriseOne Supply Chain Planning/APAg/vers_ x.x.x.
/f_ui.tcl" -display @d
```

Starting the Pack Utility in UNIX

To start the pack utility in UNIX:

- In a UNIX window, enter apag at a command line.

  A list of options appears.

- Type 5 and press Enter.

Packing Projects

When you pack a project, a directory of the same name (without the .prj extension) is created in the directory specified in the Directory field. The directory contains all the necessary files to unpack. After a project is packed into a single directory, you can tar the contents into a single file and compress the file so that it can be easily distributed.

To pack a project:

- Open the pack utility window.

  The settings of the NMX_INIT and NMX_RES environment variables are displayed in the pack utility window. You can change them if necessary.

- Click Pack.
- Click Add.
- In the Project Selector window, select the projects that you want to pack.

  As you select projects, they appear in the Projects field in the pack utility window.

- When you do not want to select any more projects, click Cancel in the Project Selector window.

  Click Drop in the pack utility window if you want to remove any projects.

- In the pack utility window, in the Directory field, enter the name of the directory into which the projects are to be packed and note the following:
- Specify a directory, not a file.
- If you specify a directory that exists, its contents are deleted.
- You can create a directory only if the path to that directory exists. For example, you can specify the directory */path/example/directory* only if */path/example* already exists.
- You can click Select to display the File Selector window in which you can select a directory.
- In the APAg Admin Password field, enter the password for the Administrator for the repository (specified by NMX_INIT and NMX_RES) from which you are packing.
- Click Proceed.

  A window appears, asking if you want to export the data for a data table. The window has four choices: Yes, Yes to all, No, or No to all. If you want to export the data, select Yes. If you do not want to export the data for that table, select No. If you want to export the data for all the data tables, select Yes to all.

  You are prompted for every data table specified in the project (that is, for every data object listed in the Data Object Manager window), even for those not actually being used in a data flow in the project. To bypass all the prompts, you can select Yes to all when prompted to export data.

  The packing process now takes place.

Note

When you pack a project, ensure that every brace bracket has a corresponding opening or closing brace bracket in the data.

Unpacking a Project

These steps assume that the directory into which a project was packed has been untarred or uncompressed.

To unpack a project:

- In the pack utility window, click Unpack.
- In the Directory field, enter the name of the directory that contains the packed project. You can click Select to find the directory. When you select a directory, the packed projects are displayed in the Projects field.
- Select the specific projects that you want to unpack.
- In the APAg Admin Password field, enter the password for the Administrator for the repository (specified by NMX_INIT and NMX_RES) from which you are unpacking.
- Click Proceed. If a storage definition cannot be found that matches the storage definition name that held the tables in the original project, you are prompted to select the storage into which the data is to be stored.

Note

If an existing table has the same name as a packed table, the table is overwritten by the packed table.

Exporting Data Using the Pack Utility

Exporting data can be useful if you want to move data out of one kind of storage and import it into another – for example, you can export data from an Oracle database at one site to flat file for use at another site.

To export data using the pack utility

- In the pack utility window, click Export.
- Select the kind of data that you want to export. The following table outlines your options:

Selection	What it exports

Selection	What it exports
Repository	Storage definitions, table definitions, and data. You can export the information for only one repository at a time (the repository specified in the NMX_INIT field).
Storage	Storage definitions (optionally), table definitions, and data.
Storage Def.	Storage definitions only.
Table	Table definitions and their data, which can come from multiple storages.
Table Def.	Table definitions only, which can come from multiple storages.

- In the APAg Admin Password field, enter the password for the administrator for the repository from which you are exporting.

  The repository is specified by the NMX_INIT and NMX_RES environment variables.

- If you selected to export Storage, Storage Def., Table, or Table Def., select Add.
- For each item that you want to export, select the item in the selector window and click Add.
- In the pack utility window, click Close.
- (Optional). Click Drop to remove an entry.
- Repeat steps 2 to 6 to export other kinds of data.
- In the Filename field, enter the name of the file into which the data is to be exported.

  You can specify an existing file that was previously exported if you want to append information to it.

- Click Proceed.

  The selected information is exported to the file that you specified.

Importing Data Using the Pack Utility

If you exported multiple items to one file, you must import them at one time.

To import data using the pack utility:

- In the pack utility window, select Import.
- In the Filename field, enter the name of the file into which the data was exported. You can click on Select to display a selection dialog to find and select the file.
- In the APAg Admin Password field, enter the password for the Administrator of the repository (specified by NMX_INT and NMX_RES) into which you are importing.
- Select Proceed.

You are prompted if you want to customize how the data is imported. You can customize all the data or on the storages. Customization windows are displayed, depending on your selection. The data is imported.

XML Support

This section provides an overview of Advanced Planning Agent support for XML and discusses how to:

- Transform relational data tables into XML documents.
- Transforming XML documents into relational data tables.

Understanding Advanced Planning Agent Support for XML

XML (eXtensible Markup Language) is a standard syntax for coding documents and messages to use on the World Wide Web. Advanced Planning Agent provides two files for processing XML files. They are:

- xmlencode.exe, which converts flat file tables to an XML file in the SCP Encoding or S2 format
- xmlparse.exe, which converts S2 and SCP Encoding XML files into flat file tables

Five files are installed with Advanced Planning Agent:

File Name	Description
xmlencode.exe	The xmlencode program transforms relational data tables to target S2 standard XML documents.
xmlparse.exe	The xmlparse program transforms source XML documents into relational data tables.
APSencoding.xsd	The SCP Encoding schema file.
s2.xsd	The S2 schema file that describes the s2 standard.
encodeCfg.xsd	The encodeCfg.xsd file describes the schema for the input parameter file that is used with xmlencode.
parseCfg.xsd	The parseCfg.xsd file describes the schema for the input parameter file that is used with xmlparse.

To use these programs, you need a basic understanding of XML, the SCP Encoding and S2 Standard XML formats, and the two configuration file formats, which are also XML. Some knowledge of XSLT is also required.

The following table defines the terms that are used in relation to XML:

Term	Definition
Relational data file	A flat file having the same number of columns in every row. The file is ASCII text. If one column contains significant spaces, its content has to be enclosed in quotes.
Schema	XML language for defining classes of XML documents.

Term	Definition
Supply Chain Planning Encoding Standard	Standard for encoding relational data files in SCP Encoding XML.
S2 Standard	Standard for encoding relational data files in S2 XML. This is an intermediate format, which facilitates conversion to an arbitrary XML document. Specific to S2 format is that the data fields themselves are wrapped with XML element identification.
XSL	eXtensible Style Language. Used for expressing style sheets. It consists of two parts: An XML vocabulary for specifying formatting semantics. An XSL application can transform XML documents to or from other classes of (XML) documents, using a transformation or formatting description. A language for transforming XML documents (XSLT).
XSLT	eXtensible Stylesheet Language Transformations. Used for transforming XML documents into other XML documents (part of XSL).
Xerces	An XML parser developed under the Apache XML Project (http://xml.appache.org/index.html).
Xalan	Applies XSL to XML documents; developed under the Apache XML Project (http://xml.appache.org/index.html).

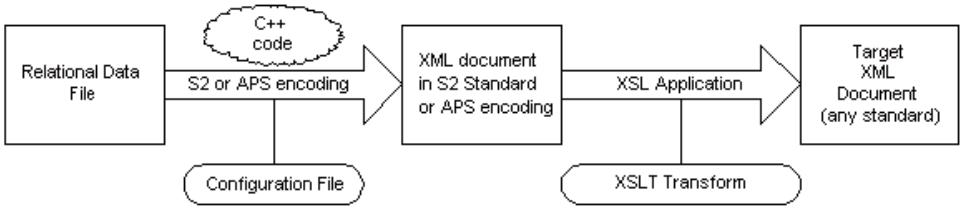
Transforming Relational Data Tables into XML Documents

This section discusses how to:

- Run the xmlencode program.
- Use run-time parameters.
- Control which configuration file parameters appear in the output file.
- Control element naming.
- Specify the output style.
- Create input relational data files.
- Create configuration files.

Running the xmlencode Program

When running the xmlencode program, parameters are passed to the encoder from an XML file. This parameter file is based on the encodeCFG.xsd schema document. The following diagram illustrates how a relational data table is transformed to an XML document:



xmlencode program process

The xmlencode program converts a relational data table into an S2 or SCP Encoding XML document. The syntax for running this program is as follows:

```
xmlencode –f <configFile> [-l <logFile>] [-param name1 value1 [… namen valuen]]
```

where:

Parameter	Description
`-f <configFile>`	Specifies the name and path of the XML configuration file. This argument is required.
`-l <logFile>`	Generates a log file of the name <log file> to store any error messages while encoding. This argument is optional. If this argument is omitted, a log file named xmllog is created in the current directory.
`-param name1 value1`	Passes the specified runtime parameter, <name1>, to the encoder with a value of <value 1>. Multiple parameters can be passed to the encoder. Note that if the arguments after `–param` are not in pairs, then an error message is generated and the encoder aborts. This argument is optional. **Note** Runtime parameters can also be passed to the encoder using a parameter file. Supply the path and name of the parameter file by using the run-timeParamFile element in the configuration file.

The following example shows the output S2 XML standard:

```
<jdeMessage Set>
  <jdeMessage name=”Table1”>
      < jdeRecord>
      <jdeField name="column1">100.508</jdeField>
      <jdeField name="column2">0</jdeField>
      <jdeField name="column3">100.508</jdeField>
      <jdeField name="column4">101.342</jdeField>
      </ jdeRecord >
      < jdeRecord>
      <jdeField name="column1">200.508</jdeField>
      <jdeField name="column2">7</jdeField>
      <jdeField name="column3">300.008</jdeField>
      <jdeField name="column4">517.315</jdeField>
      </ jdeRecord >
```

```
  </jdeMessage>
</jdeMessageSet>
```

Using Run-time Parameters

You can pass run-time parameters to the encoder from the command line or by using an XML configuration file. To specify that you want to use a configuration file, add the `run-timeParamFile` element to your configuration file immediately after the `generalParameters` element. For example:

```
<run-timeParamFile>C:\SCP\MyRunTimeParameters.xml</run-timeParamFile>
```

The encoder processes run-time parameters with the highest precedence. The encoder uses the following rules when processing configuration files:

- If the name of a run-time parameter matches the name of a general or file parameter, then the value of the run-time parameter is used in the output file.
- If the run-time parameter specified in a configuration file does not have a value, then it is ignored.
- If the name of a run-time parameter does not match any general or file parameter, then it is ignored.

Here is an example of a run-time parameter file:

```
<?xml version="1.0" encoding="utf-8"?>
<run-timeParameters>
  <param name="time-bucket">weekly</param>
  <param name="start-date">2002-15-05</param>
  <param name="count">23 </param>
</run-timeParameters>
```

Controlling Which Configuration File Parameters Appear in Output Files

You can specify which parameters from the configuration file that you want to appear in your output file by using the `outputStructure's parameters` element. The parameter names specified in the `outputStructure` element must match the name of a general or file parameter; otherwise, an error message is generated.

If the `parameters` element is missing from the `outputStructure` element, then no configuration file parameters are written to the output file. If the `outputStructure` element is missing from the description of an `inputTable`, then all of the general and file parameters are written in the <parameters> section at the beginning of the output file. If both a file and general parameter share the same name, then the parameter is only written once to the output file, using the value of the file parameter.

Controlling Element Naming

The inputTable's `namingStyle` attribute determines how elements are named when encoding to XML. The attribute can be set to either "verbose" or "explicit." The default value for the `namingStyle` attribute is "verbose."

When using verbose naming, define elements using the `<jdeField>` tag. For example, a column named "count" with the value 3, is encoded in the S2 standard as:

```
<jdeField name="count">3</jdeField>
```

When using explicit naming, the element can be labeled directly. For example, the same column mentioned above can be encoded as:

```
<count>3</count>
```

Specifying the Output Style

The encoder produces output XML documents with hierarchical structuring of data that is based on the field values of the group members. Data can be grouped in either List or Row style:

- List style groups the columns of the innermost layer into one complex element
- Row style groups the columns of the innermost layer in rows

By default, xmlencode produces List style output. How you configure the `outputStructure` element determines the type of output that is generated. Adding or omitting an empty group within the innermost group configures the output style.

List Style XML Output

With list style output, the columns of the innermost layer are grouped into one complex element, even if the list contains the values from more rows. This style is the default output.

In the following XML file example, the `<group2>` elements contain the values of the `<value>`, `<offset>`, and `<property>` columns, which have identical values for the columns that form `<group1>`. One `<group1>` element contains only one `<group2>` element.

```
<?xml version="1.0" encoding="utf-8"?>
<jdeMessageSet>
  <jdeMessage name="SampleFile">
    <parameters>
  <time-bucket>monthly</time-bucket >
  <start-date>2002-12-04</start-date >
  <count>12</count>
    </parameters>
  <group1>
      <channel>C0</channel>
      <location>L1</location>
      <product>545413</product>
    <group2>
      <value>100.508</value>
  <offset>1</offset>
  <property>AEA</property>
      <value>0</value>
  <offset>2</offset>
  <property>BEA</property>
```

```
      <value>100.508</value>
    <offset>3</offset>
    <property>CEA</property>
      </group2>
    </group1>
    <group1>
      <channel>C0</channel>
      <location>L2</location>
      <product>545422</product>
      <group2>
      <value>101.342</value>
    <offset>4</offset>
    <property>DEA</property>
      <value>98.7083</value>
    <offset>5</offset>
    <property>EEA</property>
      </group2>
    </group1>
    <group1>
      <channel>C0</channel>
      <location>L3</location>
      <product>545452</product>
      <group2>
      <value>100.508</value>
    <offset>6</offset>
    <property>FEA</property>
      <value>101.342</value>
    <offset>7</offset>
    <property>GEA</property>
      <value>117.815</value>
    <offset>8</offset>
    <property>HEA</property>
      </group2>
    </group1>
  </ jdeMessage>
  </jdeMessageSet>
```

Row Style XML Output

With row style output, the columns of the innermost layer are grouped in rows.

In the following XML file example, the complex element <group2> contains the values of the value, offset and property columns from a single row. One <group1> element contains a number of <group2> elements (one for each row in <group1>).

```
<?xml version="1.0" encoding="utf-8"?>
<jdeMessageSet>
  <jdeMessage name="SampleFile">
    <parameters>
  <time-bucket>monthly</time-bucket >
  <start-date>2002-12-04</start-date >
  <count>12</count>
    </parameters>
  <group1>
      <channel>C0</channel>
      <location>L1</location>
      <product>545413</product>
  <group2>
        <value>100.508</value>
```

```
        <offset>1</offset>
        <property>AEA</property>
        </group2>
        <group2>
        <value>0</value>
        <offset>2</offset>
        <property>BEA</property>
        </group2>
        <group2>
        <value>100.508</value>
        <offset>3</offset>
        <property>CEA</property>
        </group2>
    </group1>
    <group1>
        <channel>C0</channel>
        <location>L2</location>
        <product>545422</product>
        <group2>
        <value>101.342</value>
        <offset>4</offset>
        <property>DEA</property>
        </group2>
        <group2>
        <value>98.7083</value>
        <offset>5</offset>
        <property>EEA</property>
        </group2>
    </group1>
    <group1>
        <channel>C0</channel>
        <location>L3</location>
        <product>545452</product>
        <group2>
        <value>100.508</value>
        <offset>6</offset>
        <property>FEA</property>
        </group2>
        <group2>
        <value>101.342</value>
        <offset>7</offset>
        <property>GEA</property>
        </group2>
        <group2>
        <value>117.815</value>
        <offset>8</offset>
        <property>HEA</property>
        </group2>
    </group1>
</jdeMessage>
</jdeMessageSet>
```

Producing Row Style Output

By default, the xmlencode program produces list-style output. To tell the encoder to create a row style XML document, add an empty group to the innermost group in your configuration file. For example:

```
    <outputStructure>
        <parameters name="FileParams">
```

```
            <time-bucket/>
            <start-date/>
            <count/>
        </parameters>
        <group name="group1">
             <column>channel</column>
             <column>location</column>
             <column>product</column>
             <group name="group2">
                 <column>value</column>
                 <column>offset</column>
                 <column>property</column>
                 <group name="emptyGroup"/>
              </group>
        </group>
</outputStructure>
```

Creating Input Relational Data Files

The xmlencode input file is an ASCII file that contains a two-dimensional table structure. Flat files exported from OneWorld or created by Advanced Planning Agent are examples for the format to use. The following details must be followed when creating the file:

- Separate each row in the input file with a new line ("\n") character.
- The fields in each row correspond to the columns of a table. Separate each field with the tab ("\t") character.
- Each row must contain the same number of columns

The following example is a relational data file:

Channel	Location	Product	Value	Offset	Property
C0	L1	545413	100.508	1	AEA
C0	L1	545413	0	2	BEA
C0	L1	545413	100.508	3	CEA
C0	L1	545422	101.342	4	DEA
C0	L1	545422	98.7083	5	EEA
C0	L1	545422	100.508	6	FEA
C0	L1	545422	101.342	7	GEA
C0	L1	545422	117.815	8	HEA

Creating Configuration Files

Because of the configurable nature of the encoder, elements can be stored in an XML file and then passed to the encoder at runtime.

The encodeCfg.xsd schema describes how the input configuration file that is used with xmlencode should be formatted. It is in the \APAg\vers_x.x.x\bin directory.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:annotation>
    <xsd:documentation>
      This is an xml-schema document that describes the configuration
      file format for the xmlencode program
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="encoderConfig" type="encoderCfgType"/>

  <xsd:complexType name="encoderCfgType">
    <xsd:sequence>
      <xsd:element name="target" type="xsd:string" minOccurs="0"/>
      <xsd:element name="generalParameters" type="ParameterType"
           minOccurs="0"/>
      <xsd:element name="run-timeParamFile"  type="xsd:string"
           minOccurs="0"/>
      <xsd:element name="xmlOutput" minOccurs="0" maxOccurs="1">
        <!-- if xmlOutput is omitted the output is written to standard
          output -->
    <xsd:complexType base="xsd:string" derivedBy="extension">
          <xsd:attribute name="writeMode">
            <xsd:sympleType>
              <xsd:restriction base="xsd:string">
                <xsd:enumeration value="a"/>
                <xsd:enumeration value="w"/>
              </xsd:restriction>
            </xsd:sympleType>
          </xsd:attribute>
          <!-- writeMode can be "a" (append) or "w" (overwrite)
               "a" is the default value -->
    </xsd:complexType>
      </xsd:element>
      <xsd:element name="inputTables" type="inputTableList"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="ParameterType">
    <xsd:element name="param" maxOccurs="unbounded">
      <xsd:complexType base="xsd:string" derivedBy="extension">
        <xsd:attribute name="name" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:complexType>

  <xsd:complexType name="inputTableList">
    <xsd:element name="inputTable" type="inputTable" minOccurs="1"
      maxOccurs="unbounded"/>
  </xsd:complexType>

  <xsd:complexType name="inputTable">
    <xsd:sequence>
      <xsd:element name="tablePath" type="xsd:string"/>
      <xsd:element name="tableParameters" type="ParameterType"
           minOccurs="0"/>
      <xsd:element name="columnNames" type="columnNamesType" />
```

```
      <xsd:element name="outputStructure" type="outputStructureType"
           minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="namingStyle" default="verbose">
      <xsd:sympleType>
         <xsd:restriction base="xsd:string">
            <xsd:enumeration value="verbose"/>
            <xsd:enumeration value="explicit"/>
         </xsd:restriction>
      </xsd:sympleType>
    </xsd:attribute>
  </xsd:complexType>

  <xsd:complexType name="columnNamesType">
    <xsd:element name="column" type="xsd:string"
maxOccurs="unbounded"/>
  </xsd:complexType>

  <xsd:complexType name="outputStructureType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
  This new element in the table description describes the
  structure of XML output files. If missing, then the output
  XML conforms to the S2 schema
      </xsd:documentation>
    </xsd:annotation>

    <xsd:sequence>
      <xsd:element name="parameters" type="ParameterType"
           minOccurs="0"/>
  <!-- Under the parameters complex element enter the parameters
           that need to be included in the output file -->
      <xsd:element name="group" type="outputGroupType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="outputGroupType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
  This is a recursive element.
  If in the innermost group you specify only columns, then the
      output is produced in "list style".
  If in the innermost group you specify an empty group,
      then the output is produced in "row style".
      </xsd:documentation>
    </xsd:annotation>

    <xsd:sequence>
      <xsd:element name="column" type="xsd:string" minOccurs="0"
          maxOccurs="unbounded"/>
      <xsd:element name="group" type="outputGroupType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>

</xsd:schema>
```

Elements of xmlencode Configuration Files

The following elements are found in an xmlencode configuration file:

Element Name	Parent	Occurs	A/SE	Description
encoderConfig	root	1		The root of the parameter document list.
Target	encoderConfig	0-1	SE	Identifies the target of the message.
generalParameters	encoderConfig	0-1	SE	Groups the parameter data that applies to the entire message set.
Param	generalParameters	1	SE	Indicates a parameter value.
run-timeParamFile	encoderConfig	0-1	A	Specifies the parameter file name to be passed to the encoder. See *Using a Configuration File* for more information.
Name	param	1	A	The name of the associated parameter.
xmlOutput	encoderConfig	0-1	SE	Indicates the location of the output (XML document) of the xmlencode program.
writeMode	xmlOutput	0-1	A	Specifies the write mode: “a” for append to file “w” for overwrite (default)
inputTables	encoderConfig	1	SE	Contains the input tables.
inputTable	inputTables	1-n	SE	Specifies the relevant information that concerns a data table.
namingStyle	inputTables	0-1	A	Specifies the element naming method. This element can be set to either “verbose” or “explicit.” The default value is “verbose.” See *Specifying the Output Style* for more information.
columnNames	inputTable	1	SE	Groups the column names.
Column	columnNames	1-n	SE	Specifies the name of the corresponding field in the data table.
Name	inputTable	1	A	The name of the data table.
tablePath	inputTable	1	SE	The location of the data table.
tableParameters	inputTable	0-1	SE	Groups the parameters that are specific to the table.
Param	tableParameters	0-n	SE	Contains the parameter value.
Name	param	1	A	The name of the parameter.

Element Name	Parent	Occurs	A/SE	Description
outputStructure	inputTable	0-1	SE	Describes how the columns are grouped and nested in the output XML file that is produced by the encoder. Including this element in your configuration file tells the encoder to produce SCP Encoding XML. Omitting this element tells the encoder to produce S2 Standard XML.
parameters	outputStructure	0-1	SE	Specifies which parameters from the configuration file appear in the output.
Group	outputStructure	1	SE	Specifies how to group columns from the input file and allows nesting of one group inside another. This element is recursive. If you only specify columns in the innermost group, then the output is produced in "list style." If after enumerating the columns in the most inner group, you specify an empty group, then the output is produced in "row style."
<parameter name>	parameters	0-n	SE	Specifies a parameter name and value.
Column	group	0-n	SE	Specifies the column names to be written to the output file. Note that the columns mentioned in the output structure must be a subset of the column names that are defined in the columnNames element. The order of columns can be different from the order in the columnNames element.
Group	group	0-1	SE	Adds a nested group within the outputGroup element.
Name	group	1	A	The output group name.

Note

A/SE means attribute or subelement.

Example: xmlencode Configuration File

The following example is an XML file that is used to configure the xmlencode output. It uses a runtime parameter file, ParamFile.xml, and generates output in the SCP Encoding standard, with explicit row naming, in row style. The output file is called outRow_explicit.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<encoderConfig>
  <target>owimport</target>
  <generalParameters>
    <param name="responseType">transaction</param>
```

```
    <param name="environment"> PRD733</param>
    <param name="time-bucket">monthly</param>
    <param name="start-date">2002-12-04</param>
    <param name="count">12</param>
  </generalParameters>
  <run-timeParamFile>/ParamFile.xml</run-timeParamFile>
  <xmlOutput writeMode="w">/outRow_explicit.xml</xmlOutput>
<!-- Create output in row -  explicit style -->
  <inputTables>
    <inputTable name="Table1" namingStyle="explicit">
      <tablePath>/input_data</tablePath>
      <tableParameters>
   <param name="time-bucket">weekly</param>
   <param name="start-date">2002-08-27</param>
   <param name="count">15</param>
      </tableParameters>
      <columnNames>
    <column>Channel</column>
    <column>Location</column>
    <column>Product</column>
    <column>Value</column>
    <column>Offset</column>
    <column>Property</column>
      </columnNames>
      <outputStructure>
    <parameters>
      <time-bucket/>
          <start-date/>
          <count/>
    </parameters>
           <group name="Keys">
       <column>Channel</column>
       <column>Location</column>
       <column>Product</column>
           <group name="RowsOfData">
            <column>Value</column>
            <column>Offset</column>
            <column>Property</column>
              <group name="Dummy"/>
           </group>
         </group>
      </outputStructure>
    </inputTable>
  </inputTables>
</encoderConfig>
```

Transforming XML Documents into Relational Data Tables

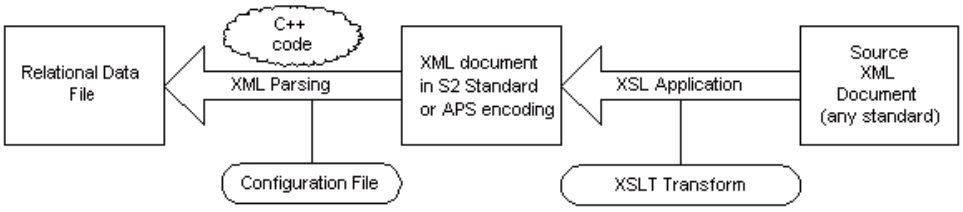
This section discusses how to:

- Run the xmlparse program.
- Create xmlparse configuration files.
- Specify S2 standard XML.

- Specify SCP encoding XML.

Running the xmlparse Program

The following diagram illustrates how an XML document is transformed into a data table:



xmlparse program process

The xmlparse program converts an SCP or S2 standard XML document into a relational data file. The syntax for running this program is:

```
xmlparse –f <configFile> [-l <logFile>]
```

where:

Argument	Description
`-f <configFile>`	Specifies the name and path of the XML configuration file. This argument is required.
`-l <logFile>`	Generates a log file of the name <log file> to store any error messages while encoding. This argument is optional.

Creating xmlparse Configuration Files

Because of the configurable nature of the parser, elements can be stored in an XML file and then passed to the parser at runtime.

The parseCfg.xsd schema describes how the input configuration file that is used with xmlparse should be formatted. It is in the \APAg\vers_x.x.x\bin directory.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <!-- This XML schema document describes the format for the
       configuration information for the S2 relational parser.
   -->
  <xsd:element name="parserConfig" type="parserConfigType"/>
  <xsd:complexType name="parserConfigType">
    <xsd:element name="sourceFile" type="xsd:string"/>
    <xsd:element name="logFile" type="xsd:string" minOccurs="0"/>
    <xsd:element name="messageSetInfo" type="messageSetInfoType"
      minOccurs="0"/>
```

```
    <xsd:element name="tableList" type="tableListType"/>
  </xsd:complexType>
  <xsd:complexType name="tableListType">
    <xsd:element name="table" type="tableType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:complexType>
  <xsd:complexType name="tableType">
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <!-- This should have a type that could only be overwrite(w)
         or append(a)
     -->
     <xsd:attribute name="writeMode" type="xsd:string" use="default"
          value="a"/>
     <xsd:element name="file" type="xsd:string">
     <xsd:element name="columnNames" type="columnNamesType" />
    </xsd:complexType>
  <xsd:complexType name="columnNamesType">
    <xsd:element name="column" type="xsd:string"
maxOccurs="unbounded">
  </xsd:complexType>
  <xsd:complexType name="messageSetInfoType">
    <xsd:attribute name="file" type="xsd:string">
    <xsd:element name="column" type="xsd:string"
maxOccurs="unbounded">
  </xsd:complexType>
</xsd:schema>
```

Elements of the xmlparse Configuration File

The elements of the xmlparse Configuration file are described in the following table:

Element Name	Parent	Occurs	A/SE	Description
parserConfig	root	1		Contains the parameter information.
sourceFile	parserConfig	1	SE	Identifies the source file that contains the XML message.
messageSetInfo	parserConfig	0-1	SE	Groups all of the output information.
File	messageSetInfo	1	A	The location of a file that can be used to store information that is specific to the message set and not to the data.
Column	messageSetInfo	1-n	SE	Specifies what data is of interest.
tableList	parserConfig	1	SE	Groups the table-specific information.
Table	tableList	0-n	SE	Groups table properties.
Name	table	1	A	Specifies the table name.
writeMode	table	0-1	A	The write mode for adding this data to the file: “a” for append “w” for overwrite
File	table	1	SE	The file location to write the table.

Element Name	Parent	Occurs	A/SE	Description
columnNames	table	1	SE	Groups the column names (order is important).
Column	columnNames	0-n	SE	Specifies the column names.
Quotable	column	0-1	A	Specifies the output format: True for quoting output False for not quoting output

Note

A/SE means attribute or subelement.

Example: xmlparse Configuration File

The following example is an XML file that is used to configure the xmlparse output. It takes an XML file, s2a.xml, as input and creates three flat files.

```
<?xml version="1.0" encoding="utf-8" ?>
<parserConfig>
  <sourceFile>c:\integration_data\october\s2ToRelational\s2a.xml
     </sourceFile>

<logFile>c:\integration_data\october\s2ToRelational\s2a.log</logFile>
  <messageSetInfo file=
     "c:\integration_data\october\s2ToRelational\s2a.msg>
      <column>targetID</column>
      <column>messageID</column>
      <!-- more columns go here. It defines what we need to include-->
      <!-- in this message file.-->
  </messageSetInfo>
  <tableList>
       <table name="F4201z1">
  <file>c:\integration_data\october\s2ToRelational\f4201z1.rel</file>
       <columnNames>
            <column>EdiUserId</column>
            <column>EdiBatchNumber</column>
            <column>EdiTransactNumber</column>
            <column>EdiLineNumber</column>
            <column>EdiDocumentType</column>
            <column>TypeTransaction</column>
            <column>EdiTranslationFormat</column>
            <column>EdiTransmissionDate</column>
            <!-- more columns go here. It defines what we need to-->
            <!-- include in this message file.-->
      </columnNames>
      </table>
    <table name="F4211z1" writeMode="a">

<file>c:\integration_data\october\s2ToRelational\f4211z1.rel</file>
        <columnNames>
            <column>EdiUserId</column>
            <column>EdiBatchNumber</column>
            <column>EdiTransactNumber</column>
```

```
            <column>EdiLineNumber</column>
            <!-- more columns go here. It defines what we need to-->
            <!-- include in this message file.-->
          </columnNames>
          </table>
    <table name="F49211z1" writeMode="w">

<file>c:\integration_data\october\s2ToRelational\f49211z1.rel</file>
          <columnNames>
          <column>EdiUserId</column>
          <column>EdiBatchNumber</column>
          <column>EdiTransactNumber</column>
          <column>EdiLineNumber</column>
          <column>EdiDocumentType</column>
          <!-- more columns go here. It defines what we need to-->
          <!-- include in this message file.-->
          </columnNames>
      </table>
  </tableList>
  </parserConfig>
```

Specifying the S2 Standard XML Schema

The S2 XML standard is a standard where each row from an input table is encoded in a <jdeRecord> element that contains <jdeField> elements. The following example represents the S2 XML schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:annotation>
    <xsd:documentation>
      This is an xml-schema document that describes the S2 standard
     format for table conversion to XML.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="jdeMessageSet" type="jdeMessageSetType"
    minOccurs="0" maxOccurs="unbounded"/>
  <xsd:complexType name="jdeMessageSetType">
    <xsd:element name="messageID" type="xsd:string" minOccurs="0"/>
    <xsd:element name="jdeTarget" type="xsd:string" minOccurs="0"/>
    <xsd:element name="jdeDocParams" type="jdeParamsType"
          minOccurs="0"/>
    <xsd:element name="jdeMessage" type="jdeMessageType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:complexType>
  <xsd:complexType name="jdeMessageType">
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:element name="jdeDocDesc" type="jdeDocDescType"
          minOccurs="0"/>
    <xsd:element name="jdeDocParams" type="jdeParamsType"
          minOccurs="0"/>
    <xsd:element name="jdeRecord" type="jdeRecordType" minOccurs="0"
          maxOccurs="unbounded"/>
  </xsd:complexType>
  <xsd:complexType name="jdeDocDescType">
    <xsd:element name="jdeDocDefn" type="jdeDocDefnType"
          minOccurs="0"/>
  </xsd:complexType>
```

```
  <xsd:complexType name="jdeParamsType">
    <xsd:element name="jdeParam" maxOccurs="unbounded">
      <xsd:complexType base="xsd:string" derivedBy="extension">
        <xsd:attribute name="name" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:complexType>
  <xsd:complexType name="jdeRecordType">
    <xsd:element name="jdeField" maxOccurs="unbounded">
      <xsd:complexType base="xsd:string" derivedBy="extension">
        <xsd:attribute name="name" type="xsd:string"/>
        <xsd:attribute name="nullFlag" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:complexType>
</xsd:schema>
```

Elements of the S2 Standard XML Schema

The following elements are found in the S2 Standard XML schema:

Element Name	Parent	Occurs	A/SE	Description
jdeMessageSet	Root	0-n		Groups multiple tables into a single message transaction.
messageId	jdeMessageSet	0-1	SE	Identifies the message for future reference.
jdeTarget	jdeMessageSet	0-1	SE	Identifies the destination or destinations of the message.
jdeDocParams	jdeMessageSet	0-1		Groups the parameter data that applies to the entire message set. Parameter data is extra data that is needed to create the target XML document but is not part of the relational data.
jdeParam	jdeDocParams	0-n		Indicates a parameter value.
Name	jdeParam	1	A	The name of the associated parameter.
jdeMessage	jdeMessageSet	0-n	SE	Contains the data records from a relational table.
Name	jdeMessage	1	A	The name of the relational table.
jdeDocParams	jdeMessage	0-1	SE	Groups the parameters that apply to a specific table.
jdeParam	jdeDocParams	0-n	SE	Indicates a parameter value.
Name	jdeParam	1	A	The name of the associated parameter.
jdeRecord	jdeMessage	0-n	SE	Groups the fields of a given record from the relational data table.
jdeField	jdeRecord	0-n	SE	Indicates the contents of a field.

Element Name	Parent	Occurs	A/SE	Description
Name	jdeField	1	A	The name of the field.
nullFlag	jdeField	0-1	A	Indicates if the field is null. Valid values are: TRUE FALSE If neither value is present, then the field is non-null.

Example: S2 Standard XML Document

The following example is a S2 Standard-formatted XML document:

```
<?xml version="1.0" encoding="utf-8"?>
<jdeMessageSet>
  <messageID>0x753a</messageID>
  <jdeTarget>owimport</jdeTarget>
  <jdeDocParams>
    <jdeParam name="responseType">transaction</jdeParam>
    <jdeParam name="user">JDE</jdeParam>
    <jdeParam name="environment">PRD733</jdeParam>
    <jdeParam name="transactionType">JDESOOUT</jdeParam>
    <jdeParam name="action">transactionInfo</jdeParam>
    <jdeParam name="returnCode">0</jdeParam>
    <jdeParam name="returnDesc">XML Request OK</jdeParam>
  </jdeDocParams>
  <jdeMessage name="keydata">
    <jdeRecord>
      <jdeField name="EdiUserId">JDE</jdeField>
      <jdeField name="EdiBatchNumber">49740</jdeField>
      <jdeField name="EdiTransactNumber">139581</jdeField>
    </jdeRecord>
  </jdeMessage>
  <jdeMessage name="F4201z1">
    <jdeDocParams>
      <jdeParam name="tableName">F4201Z1</jdeParam>
      <jdeParam name="infoType">header</jdeParam>
    </jdeDocParams>
    <jdeRecord>
      <jdeField name='EdiUserId'>JDE</jdeField>
      <jdeField name='EdiBatchNumber'>49740</jdeField>
      <jdeField name='EdiTransactNumber'>139581</jdeField>
      <jdeField name='EdiLineNumber'>1.000</jdeField>
      <jdeField name='EdiDocumentType'>SO</jdeField>
      <jdeField name='TypeTransaction'>JDESOOUT</jdeField>
      <jdeField name='EdiTranslationFormat' isNull='TRUE'> </jdeField>
      <jdeField name='EdiTransmissionDate'  isNull='TRUE'></jdeField>
    </jdeRecord>
  </jdeMessage>
    <jdeMessage name="F4211z1">
    <jdeDocParams>
      <jdeParam name="tableName">F4211Z1</jdeParam>
      <jdeParam name="infoType">detail</jdeParam>
    </jdeDocParams>
    <jdeRecord>
```

```
      <jdeField name="EdiUserId">JDE</jdeField>
      <jdeField name="EdiBatchNumber">49740</jdeField>
      <jdeField name="EdiTransactNumber">139581</jdeField>
      <jdeField name="EdiLineNumber">3.000</jdeField>
      <jdeField name="EdiDocumentType">SO</jdeField>
      <jdeField name="TypeTransaction">JDESOOUT</jdeField>
      <jdeField name="EdiTranslationFormat" isNull=”TRUE”> </jdeField>
      <jdeField name="EdiTransmissionDate" isNull=”TRUE”></jdeField>
    </jdeRecord>
    <jdeRecord>
      <jdeField name="EdiUserId">JDE</jdeField>
      <jdeField name="EdiBatchNumber">49740</jdeField>
      <jdeField name="EdiTransactNumber">139581</jdeField>
      <jdeField name="EdiLineNumber">2.000</jdeField>
      <jdeField name="EdiDocumentType">SO</jdeField>
      <jdeField name="TypeTransaction">JDESOOUT</jdeField>
      <jdeField name="EdiTranslationFormat" isNull=”TRUE”> </jdeField>
      <jdeField name="EdiTransmissionDate" isNull=”TRUE”></jdeField>
    </jdeRecord>
    <jdeRecord>
      <jdeField name="EdiUserId">JDE</jdeField>
      <jdeField name="EdiBatchNumber">49740</jdeField>
      <jdeField name="EdiTransactNumber">139581</jdeField>
      <jdeField name="EdiLineNumber">1.000</jdeField>
      <jdeField name="EdiDocumentType">SO</jdeField>
      <jdeField name="TypeTransaction">JDESOOUT</jdeField>
      <jdeField name="EdiTranslationFormat" isNull=”TRUE”> </jdeField>
      <jdeField name="EdiTransmissionDate" isNull=”TRUE”></jdeField>
    </jdeRecord>
  </jdeMessage>
  <jdeMessage name="F49211z1">
    <jdeDocParams>
      <jdeParam name="tableName">F49211Z1</jdeParam>
      <jdeParam name="infoType">additionalHeader</jdeParam>
    </jdeDocParams>
  </jdeMessage>
</jdeMessageSet>
```

Specify Supply Chain Planning Encoding XML

The following XML file represents the SCP encoding schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <xsd:annotation>
    <xsd:documentation>
      This is an xml-schema document that describes the "SCP encoding"
      standard
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="jdeMessageSet" type="jdeMessageSetType"
    minOccurs="0" maxOccurs="unbounded"/>

  <xsd:complexType name="jdeMessageSetType">
    <xsd:element name="messageID" type="xsd:string" minOccurs="0"/>
```

```
    <xsd:element name="jdeTarget" type="xsd:string" minOccurs="0"/>
    <xsd:element name="jdeMessage" type="jdeMessageType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:complexType>

  <xsd:complexType name="jdeMessageType">
    <xsd:sequence>
      <xsd:element name="parameters" type="jdeParamsType"
          minOccurs="0"/>
      <xsd:element type="jdeObjectType" minOccurs="0"
          maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="jdeParamsType">
    <xsd:element base="xsd:string" maxOccurs="unbounded"/>
  </xsd:complexType>

  <xsd:complexType name="jdeObjectType">
  <xsd:element type="groupType" maxOccurs="unbounded"/>
  </xsd:complexType>

  <xsd:complexType name="groupType">
    <xsd:sequence>
      <xsd:element type="jdeFieldType" maxOccurs="unbounded"/>
      <xsd:element type="groupType" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="jdeFieldType">
      <xsd:complexType base="xsd:string" derivedBy="extension">
        <xsd:attribute name="name" type="xsd:string"/>
        <xsd:attribute name="isNull" type="xsd:string"/>
    <!-- Valid values for "isNull" attribute are "true" and "false" --
>
      </xsd:complexType>
  </xsd:complexType>

</xsd:schema>
```

Elements of the SCP Encoding XML Schema

The following elements are found in the SCP Encoding XML schema:

Note

If the namingStyle attribute is set to “explicit,” you must provide names for the <jdeField> and <jdeGroup> elements.

Element Name	Parent	Occurs	A/SE	Description
jdeMessageSet	Root	0-n		Groups multiple tables into a single message transaction.
messageId	jdeMessageSet	0-1	SE	Identifies the message for future reference.

Element Name	Parent	Occurs	A/SE	Description
jdeTarget	jdeMessageSet	0-1	SE	Identifies the destination or destinations of the message.
jdeMessage	jdeMessageSet	0-n	SE	Contains the data records from a relational table.
parameters	jdeMessage	0-n		Indicates a parameter value.
Name	jdeMessage	1	A	The name of the relational table.
jdeGroup or <group name>	jdeMessage	1-n	SE	Groups the fields of a given record from the relational data table. This element is recursive. Groups can also be nested within fields.
jdeField or <field name>	jdeGroup	0-n	SE	Indicates the contents of a field.
Name	jdeField	1	A	The field name.
isNull	jdeField	0-1	A	Indicates if the field is null. Valid values are: TRUE FALSE If neither value is present, then the field is non-null.

Example: SCP Encoding XML Document

The following example is an SCP Encoding XML document:

```
<?xml version="1.0" encoding="utf-8"?>
<jdeMessageSet>
  <jdeMessage name="SampleFile">
    <parameters>
  <time-bucket>monthly</time-bucket >
  <start-date>2002-12-04</start-date >
  <count>12</count>
    </parameters>
  <group1>
      <channel>C0</channel>
      <location>L1</location>
      <product>545413</product>
    <group2>
      <value>100.508</value>
  <offset>1</offset>
  <property>AEA</property>
      <value>0</value>
  <offset>2</offset>
  <property>BEA</property>
      <value>100.508</value>
  <offset>3</offset>
  <property>CEA</property>
    </group2>
  </group1>
```

```
  <group1>
      <channel>C0</channel>
      <location>L2</location>
      <product>545422</product>
    <group2>
      <value>101.342</value>
  <offset>4</offset>
  <property>DEA</property>
      <value>98.7083</value>
  <offset>5</offset>
  <property>EEA</property>
    </group2>
  </group1>
  <group1>
      <channel>C0</channel>
      <location>L3</location>
      <product>545452</product>
    <group2>
      <value>100.508</value>
  <offset>6</offset>
  <property>FEA</property>
      <value>101.342</value>
  <offset>7</offset>
  <property>GEA</property>
      <value>117.815</value>
  <offset>8</offset>
  <property>HEA</property>
    </group2>
  </group1>
</ jdeMessage>
</jdeMessageSet>
```

Index

ABS, 193
ABSDATE, 196
access, 48
ACOS, 194
Administrator, 9
  starting in Windows, 14, 15
Advanced Planning Agent
  Administrator, 9
  Data Flow Launcher, 9
  Manager, 9
Advanced Planning Agent functions, 178
ADVANCEDAYS, 197
aggregation filter node, 85, 86, 88, 89, 90, 93, 98
aligning nodes in data flows, 66
arcs, 57, 66, 140
ASCIICONVERT, 199
ASIN, 200
ASSIGN, 200
ATAN, 201
ATAN2, 201
attach points, 57, 59
AVG, 202
BACKGROUND, 203
batch files, 126, 128, 130
  running, 130
BETWEEN, 203
BLACK, 204
BLUE, 205
browse permissions, 49
browsing data, 74
bucketizing, 96
bucketizing data, 93
BUCKETLENGTHS, 93, 205
BUCKETPERIODS, 93, 206
BUCKETVALUES, 93, 206
building projects, 11
business process, 5
changes only option, 71
changing
  log location, 137
  maximum log size, 138
  truncate value of logs, 138
clearing
  error indicators, 133
  run information, 140
color functions, 187
comments, 58
compressing error messages, 135
configuration files, 20, 24, 307
  xmlencode, 312
  xmlparse, 318
COS, 207
COSH, 207
COVER, 208
creating
  data object definitions, 28, 29, 30
  nodes, 65
  projects, 52
  storage definitions, 26
  xml files, 306
creating batch files, 126, 128
creating subflows, 62
Data Editor, 74, 116, 117, 139, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 162, 163, 164, 165, 166, 169, 170, 171, 172, 173, 174, 175
  opening, 151, 152
  overview, 151
Data Editor functions, 178
Data Editor nodes, 116, 118
data file
  transforming into XML, 305
Data Flow Launcher, 9, 127
data flows, 59, 63, 66, 116, 123, 126, 128, 129, 132, 133, 141, 148
data object definitions, 28, 29, 30, 31, 32
data object field definitions, 32
data object nodes, 68, 70, 73, 74
Data Object nodes, 117
data objects, 53
data types, 38, 40, 44, 45, 140
database tables, 175
databases, 37
  connecting to, 41
date functions, 185
DATETOUMTSEC, 211
DAY, 212
DAYDUR, 213
DB2, 41
definitions
  RelOffset, 267
delete matched option, 72
delete/write option, 70
deleting
  data object definitions, 31
  parameters, 141

Depletion, 214
documenting data flows, 58
editing
  parameters, 141
editing data, 74
environment variables, 131
EQTOL, 216
ERROR, 216
error indicators, 133
error messages, 133, 134, 135, 136
examples, 77, 89, 105
Examples, 34
exception output, 73
EXP, 217
exporting data
  pack utility, 302
expression editor, 171
expressions, 81, 162, 163, 164, 165, 166, 170, 171
FALSE, 217
files
  configuration, 21, 24
filter node, 86
filter nodes, 77, 84, 85, 86, 98, 101, 102, 103
FINDNEXTDAY, 217
foreign keys, 39
functions, 76, 81, 84, 88, 164, 169, 170, 171
  ABS, 193
  ABSDATE, 196
  ACOS, 194
  Advanced Planning Agent, 178
  ADVANCEDAYS, 198
  ASCIICONVERT, 199
  ASIN, 200
  ATAN, 201
  ATAN2, 201
  AVG, 202
  BACKGROUND, 203
  BETWEEN, 203
  BLACK, 204
  BLUE, 205
  BUCKETLENGTHS, 205
  BUCKETPERIODS, 206
  BUCKETVALUES, 206
  color, 187
  COS, 207
  COSH, 207
  COVER, 208
  Data Editor, 178
  date, 185
  DATETOUMTSEC, 211
  DAY, 212
  DAYDUR, 213
  Depletion, 214
  EQTOL, 216
  ERROR, 216
  EXP, 217
  FALSE, 217
  FINDNEXTDAY, 217
  GAMMA, 218
  GETDATE, 219
  GETDAYSINMONTH, 220
  GETENV, 221
  GETOL, 221
  GREEN, 222
  GTTOL, 223
  HOUR, 223
  HOURDUR, 224
  IF, 225
  INDEX, 226
  INFINITESIMAL, 227
  INFINITY, 227
  INSERT, 228
  INT, 229
  InvCover, 229
  InverseDepletion, 230
  ISNULL, 231
  Keepblanks, 231
  KillBlanks, 232
  LETOL, 233
  LN, 233
  LOG, 233
  LTRIMBLANKS, 234
  LTTOL, 235
  MAKEDATE, 235
  MAKEDATEFROMWEEK, 237, 238
  MAKEDATEWEEKFROMWEEK, 238
  MAKESYSKEY, 240
  MAX, 240
  MIN, 241
  MINUTE, 242
  MINUTEDUR, 243
  MONTH, 244
  NEG, 246
  NETOL, 246
  NEWSYSKEY, 246
  Next, 247
  NEXTMONTH, 248
  NEXTYEAR, 250
  NOT, 250
  NOW, 251
  NULL, 252

NULLDEF, 252
ORANGE, 252
OVERUNDER, 253
PARSESTRING, 254
PERCENT, 253
PI, 255
PINK, 255
POS, 256
POWER, 256
PREV, 256
PRODUCT, 257
R_AVG, 258
R_COVER, 259
R_MAX, 261
R_MIN, 262
R_PREV, 263
R_PRODUCT, 264
R_SUM, 265
RAND, 265
RED, 266
reference, 189
REGEXPR, 266
REMAINDER, 269
REMOVE, 270
REPLACEINDEX, 271
REPLACEPATTERN, 272
rolling, 188
ROUND, 273
RTRIMBLANKS, 273
SECOND, 274
SECONDUR, 276
SEQUENCE, 277
SIGN, 277
SIN, 278
SINH, 278
SQRT, 279
STDDEV, 279
string, 188
STRINGCAST, 279
STRINGCONCAT, 280
STRINGLENGTH, 281
STRINGTODOUBLE, 282
STRINGTOINT, 283
SUBSTRING, 284
SUM, 285
TAN, 286
TANH, 287
TOLOWER, 287
TOUPPER, 288
trigonometric, 187
TRIMBLANKS, 289
TRUE, 290
VIOLET, 291
WEEKDAY, 291
WEEKDUR, 293
WEEKNUMBER, 294
YEAR, 295
YEARDAYNUMBER, 296
YELLOW, 297
Functions
ASSIGN, 200
ZERO, 298
functions assign, 164
GAMMA, 218
GETDATE, 219
GETDAYSINMONTH, 220
GETENV, 221
GETOL, 221
GREEN, 222
grouping information, 86
GTTOL, 223
highlighting, 166
error messages, 136
HOUR, 223
HOURDUR, 224
IF, 225
ignore criteria option, 72
implementation, 6
implementing projects, 11
importing data
pack utility, 303
INDEX, 226
INFINITESIMAL, 227
INFINITY, 227
INSERT, 228
INT, 229
integrating systems, 8
integrations, 5
interfacing with
DB2, 41
Microsoft SQL Server, 45
Oracle, 43
InvCover, 229
InverseDepletion, 230
ISNULL, 231
join criteria, 102
join filter node, 101, 102, 103, 104, 105
join filter nodes, 107
KeepBlanks, 231
KillBlanks, 232
LETOL, 233
list style XML, 308

LN, 233
LOG, 233
log file, 135, 136
log output, 137
logs
  location, 137
  size, 138
  truncate value, 138
LTRIMBLANKS, 234
LTTOL, 235
MAKEDATE, 235
MAKEDATEFROMWEEK, 237, 238
MAKESYSKEY, 240
Manager, 9
  starting in UNIX, 51
  starting in Windows, 51
managing
  logs, 137
  user access, 48
  user permissions, 47
managing browse permissions, 49
mapping data types, 40, 140
matching patterns, 53
MAX, 240
maximum log size, 138
Microsoft SQL Server, 45
MIN, 241
MINUTE, 242
MINUTEDUR, 243
MONTH, 244
moving
  nodes, 66
NEG, 246
NETOL, 246
NEWSYSKEY, 246
Next, 247
NEXTMONTH, 248
NEXTYEAR, 250
nodes, 57, 59, 68
  aligning, 66
  connecting with arcs, 66
  creating, 65
  moving, 66
  Production Scheduling - Process, 109
NOT, 250
NOW, 251
NULL, 252
NULLDEF, 252
opening
  projects, 52
Oracle, 43, 45
ORANGE, 252
output data, 77
overriding
  parameters, 141
OVERUNDER, 253
overview, 5
overwrite option, 70
pack utility, 300, 301, 302, 303
  importing data, 303
  starting in UNIX, 300
  starting in Windows, 299
packing projects, 300
parameters, 141, 142
PARSESTRING, 254
passing environment variables to scripts, 131
PERCENT, 253
permissions, 47, 49, 50
PI, 255
PINK, 255
POS, 256
POWER, 256
PREV, 256
printing
  data object definitions, 32
  reports, 139
printing data, 116
printing reports, 174
PRODUCT, 257
Production Scheduling - Process, 24, 139
Production Scheduling - Process, 144
Production Scheduling - Process node, 145
Production Scheduling - Process node, 144, 145
Production Scheduling - Process nodes, 109, 110
projects
  creating, 52
  implementing, 11
  opening, 52
  packing, 300
  unpacking, 301
PSI editor, 144
R_AVG, 258
R_COVER, 259
R_MAX, 261
R_MIN, 262
R_PREV, 263
R_PRODUCT, 264
R_SUM, 265
RAND, 265
read before write option, 68
RED, 266
refreshing error messages, 136

REGEXPR, 266
RelOffset, 267
REMAINDER, 269
REMOVE, 270
REPLACEPATTERN, 272
reports
  printing, 139
repositories, 16, 19, 130
reset sort keys, 87
resource areas, 130
rolling function sort, 86
rolling functions, 188
rolling sort keys, 88
ROUND, 273
row style XML, 310
RPLACEINDEX, 271
RTRIMBLANKS, 273
run information, 140
running
  batch files, 130
running batch files, 126, 128, 130
running data flows, 123, 126, 128, 129, 132, 133
saving
  data object definitions, 31
saving data flows, 148
scripts, 120, 131
SECOND, 275
SECONDDUR, 276
selection criteria, 76, 79, 86, 88, 101, 102, 117, 142
selection criteria option, 72
selection filter node, 77, 79, 83
SEQUENCE, 277
shell nodes, 120, 121
shell scripts, 120
SIGN, 277
SIN, 278
SINH, 278
smart definition file, 144, 145
SQRT, 279
starting
  Administrator in Windows, 14, 15
  Manager in UNIX, 51
  Manager in Windows, 51
  pack utility in UNIX, 299
  pack utility in Windows, 299
STDDEV, 279
storage connections, 53, 55, 57
storage definitions, 25, 26
Strategic Network Optimization, 24, 139
Strategic Network Optimization nodes, 114
Strategic Network Optimization nodes, 112, 113
string functions, 188
STRINGCAST, 279
STRINGCONCAT, 280
STRINGLENGTH, 281
STRINGTODOUBLE, 282
STRINGTOINT, 283
subflows, 60, 62, 67
SUBSTRING, 284
SUM, 285
superflows, 60
Supply Chain Planning encoding XML, 325
symbols, 169
syntax rules, 81
system repositories, 16, 19
TAN, 286
TANH, 286
Tcl, 147, 148, 149
Tcl scripts, 120
TOLOWER, 287
tool command language, 147, 148, 149
Tool Command Language, 149
TOUPPER, 288
trigonometric functions, 187
TRIMBLANKS, 289
TRUE, 290
truncate value of logs, 138
unpacking projects, 301
user access, 48
user permissions, 47
verifying data flows, 63
versions, 130
view box, 63
viewing
  error messages, 133, 134, 136
viewing data, 74
VIOLET, 291
WEEKDAY, 291
WEEKDUR, 292
WEEKNUMBER, 294
wildcard characters, 53
xml files
  creating, 306, 307
  list style, 308
  row style, 310
  transforming into relational data, 317
XML files
  Supply Chain Planning encoding, 325
XML support, 304
xmlencode program, 305, 308, 312
xmlparse program, 317, 318

YEAR, 295
YEARDAYNUMBER, 296
YELLOW, 297
Z dimension, 154
ZERO, 298