

Oracle® Integrated Operational Planning, Fusion Edition

User's Guide

RELEASE 11.1.2.1

Integrated Operational Planning User's Guide, 11.1.2.1

Copyright © 2004, 2011, Oracle and/or its affiliates. All rights reserved.

Authors: EPM Information Development Team

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited. The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

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

U.S. GOVERNMENT RIGHTS:

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

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

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

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

Contents

Documentation Accessibility	13
Chapter 1. Getting Started	15
Integrated Operational Planning Analysis Process	15
Integrated Operational Planning Data Collection Process	17
Analysis Roles and Workflows	18
About Analysis Roles and Workflows	18
Analysis Owner	18
Participant	20
Approver	21
Data Provider	21
Administrator	22
Using Integrated Operational Planning	22
Logging in to Integrated Operational Planning	22
Understanding the Integrated Operational Planning User Interface	23
Chapter 2. Using the Planning Workbench	25
Planning Workbench Pages	25
Home	25
Analysis Workbench	27
Review Exceptions	27
Review Reports	28
Manage Tasks	28
Scripts	28
Statistical Forecasts	28
User Profile	28
Working with Scenarios	29
Viewing Scenarios	29
Creating Scenarios	31
Editing Scenarios	32
Undoing Changes	33
Merging Changed Entries	34

Copying Scenarios	35
Comparing Scenarios	35
Deleting Scenarios	35
Submitting Scenarios for Approval	36
Approving Scenarios	38
Completing Scenarios	38
Printing Scenarios	39
E-mailing Scenarios	40
E-mailing an Excel Shortcut to a Scenarios Planning Workbook	40
Adding Scenarios to Favorites	40
Working with Data Providers	41
About Data Providers	41
About Data Collection Scenarios	41
Assigning Data Providers to a Data Collection Scenario	42
Sending Workbooks to Data Providers	42
Monitoring Data Provider Status	43
Using and Customizing Search Filters	45
Working with Exceptions	46
About Exceptions	46
Reviewing Exceptions	46
Managing Tasks	47
Viewing Tasks	47
Creating Tasks	48
Editing Tasks	49
Deleting Tasks	50
Reviewing Reports	50
Setting Access to View Reports	51
Performing an Analysis	52
Opening a Scenario in Excel	52
Viewing Dimensional Data	59
Changing and Recalculating Measure Values	63
Reviewing the Impact of Plan Changes	68
Adding Cell Comments	69
Viewing a Scenario as Another Analysis Type	70
Refreshing Scenario Data	71
Working with Scripts	71
Working with Statistical Forecasts	72

Chapter 3. Using the Administration Workbench	77
About the Administration Workbench	77
Using the Model Tab	78
Understanding Cubes, Dimensions, and Measures Data Types	78
Working with Measure and Constraint Formulas	82
Data Flow and Mapping	85
Understanding the Model Tab User Interface	86
Building a Model	87
Reviewing and Publishing Model Objects	108
Changing Models After Publishing	109
Using the Presentation Tab	111
Managing Analysis Types	111
Managing Queries	116
Managing Report Templates	118
Reviewing and Publishing Model Objects	119
Changing Models After Publishing	120
Managing Workbooks	121
Managing Worksheets	122
Using the Administration Tab	127
Managing Essbase Connections	127
Managing the Job Queue	128
Managing the Script Editor	131
Managing Script Templates	133
Managing Security Filters	134
Managing System and Application Resources	136
Managing Users and Groups	137
Appendix A. Isadmin Commands	139
About Isadmin Commands	139
Command Options	139
Common Activities	140
Common Usage	140
Isadmin Commands and XML Files	142
XML Schema Files	142
XML Files and Search Path	142
Using Isadmin Commands in Integrated Operational Planning	143
Isadmin Commands	145
Macro	145
Schema	146

Import	147
Publish	148
Loading Members into Dimensions and Building Hierarchies	150
Export	152
Generate	153
Load	154
Sandbox Calculation, Submit, and Reconcile	157
Script Templates and JACL Scripts	158
System	160
Export MultiDimensional Data	162
Custom Java Code Commands	162
System Utility	163
Scripts in install-root/bin	164
Process Flow Modeling	164
Evolution of the Commands Through Releases	165
Scripts	165
JACL Scripts	165
Perl Module Scripts	167
Daily Scripts	167
Weekly Scripts	168
Migration Scripts	168
Appendix B. Writing Formulas	171
Keywords	171
Literals (Constants)	172
Number Literals	172
Boolean Literals	172
String Literals	173
Dimension Literals	173
Member Literals	173
Dimension Hierarchy Literals	173
Hierarchy Level Literals	174
Variables	174
Operators	175
OnChange Instructions	176
Formulas for Defining Measures	178
Formulas in Constraint Definitions	182
Handling Null (Empty) Cells	183

Appendix C. Functions	187
Basic Functions	188
abs	189
add	189
and	189
avg	189
ceiling	189
div	189
doubleValue	190
eq	190
floor	190
ge	190
getPositive	190
gt	190
isBlank	190
isNull	191
isNullString	191
isZero	191
le	191
lt	191
max	191
min	192
mul	192
ne	192
not	192
nval	192
or	192
pct	193
pctof	193
random	193
round	193
streq	193
string	193
stringValue	193
sub	193
sum	194
toBoolean	194
toDouble	194
Mapping Functions	194

Time Dimension Mapping Functions	194
Dimension Member Attribute Name Mapping Functions	195
Member Name Mapping Functions	197
Row Source Mapping Functions	197
Splicing and Splitting Mapping Functions	199
Math Functions	199
cos	199
exp	200
factorial(int x)	200
gcd	200
inflatedValue	200
log	200
log10	200
mod	200
power	201
sin	201
sqrt	201
tan	201
Member Functions	202
Defining Member Functions	202
childMembers(location hierarchy)	203
children	203
count	204
cousinLocations	204
cube	205
currentValue	205
dateMemberProperty	205
dateProperty	205
descendants	205
dimension	205
dimensionMember	206
doubleProperty	206
equals	206
exists	206
firstChild	206
firstDescendant	207
hierarchy	207
hierarchyLevel	207
in	207

indexOf	208
is Functions	208
lastChild	209
lastDescendant	210
location.....	210
lookup functions	211
member	214
memberName	214
nextMember	215
memberProperty	215
parent	215
parentMembers	216
parents.....	216
pastCousinLocations	216
previousMember	217
property.....	217
range	218
siblingCount	218
siblingLocations	219
Shape Functions	220
all	220
current.....	220
first	220
lag	221
last	221
lead	221
level	221
leveln.....	222
levels	222
moduloNext	222
moduloPrevious	222
next	222
nonLeaves	222
offset	223
previous.....	223
range	223
siblings.....	223
singleMember	224
Time Functions	227

currentTime	227
currentTimeMember	227
futureSiblingsCount	228
getDateForCurrentTimeMember	228
isCurrent	228
isCurrentDateInRange	228
isEndofPeriod	229
isFuture	229
isInRange	229
isInRangeFromCurrentTime	229
isLeftRightLeaf	230
isNext	230
isPast	231
isPrevious	231
isStartOfPeriod	232
offsetFromCurrent	232
overlapMembers	232
pastSiblingsCount	232
timeRange	232
External Functions	232
Custom Functions	233
Appendix D. MDX Extensions	235
Reverse(<i>set</i>)	235
RSQLGenerateSet(<i>RSQLQuery</i> [, <i>dimHierarchy</i> , <i>NameColumn</i> , <i>NameSpaceColumn</i>]+)	235
RowSourceLookup(<i>RowSourceColumn</i> [, <i>tuple</i>])	236
SystemPeriod(<i>level</i>)	236
Appendix E. Load XML Specifications	237
About Load XML Specifications	237
Elements	240
<delimited-file>	240
<fixed-width-file>	242
<ms-excel-file>	243
<database>	244
<database> / <query>	245
<flat-file> / <field>	245
<ms-excel-file> / <field>	247
<field> / <translate>	248

<field> / <range>	249
<flat-file> / <pivot>	249
<excel-file> / <pivot>	251
<stage-dimension>	252
<stage-dimension> / <member>	254
<stage-dimension> / <member> / <map>	255
<stage-calendar>	256
<stage-rowsource>	256
<stage-rowsource> / <map>	257
<stage-bom>	258
<stage-bom> / <map>	259
<stage-user>	259
<stage-user> / <map>	260
<stage-table>	261
<stage-table> / <map>	261
<stage-special>	262
<stage-special> / <param>	262
<prologue> <epilogue>	263
(<prologue> <epilogue>) / <param>	264
<simple-filter>	264
<dimension-filter>	266
<rowsource-filter>	267
<distinct-filter>	268
<custom-filter>	269
Value Functions	269
\${name}	269
\$add(lhs, rhs)	270
\$begin	270
\$beginmember(qname, offset)	271
\$coalesce(arg1, ..., argN)	271
\$date(string, format)	272
\$datetime	272
\$quarter	273
\$quartername	274
\$lowercase(field)	274
\$map(field, test?a:b)	274
\$mondaymonth	275
\$month	275
\$mul(lhs, rhs)	275

\$parent (dimension, hierarchy, namespace, member)	276
\$rowsource (rowsource, column, key1, expr1, ..., keyN, exprN)	276
\$rs substr (string, rbegin, rend)	276
\$seq	277
\$subfield (field, delimiterChar, index)	277
\$substr (s, begin, end)	277
\$today	278
\$trim (expr)	278
\$uppercase (expr)	278
\$week	278
Properties	279
Appendix F. Block Cache Estimation	281
Parameters	281
Equation	281
Example	281
Dense Dimensions and Java Heap Size	282
Appendix G. Troubleshooting	283
Configuration Issues	283
Properties Files (from least to highest)	283
Defaults	283
Customer Specific	284
System and Environment Issues	284
Out of Memory Error	284
Database Issues	284
E-mail Issues	285
The Server is Not Listening on the Port	285
Connection Issues	285
Deleting Objects	285
Sizing	285
Load Commands	286
Excel-Related Issues	286
Installing the Anteros Ambassador Active-X Control for Excel	286
Launching Excel When You Are Not Connected to the Internet	287
Loading From an Excel File	287
Troubleshooting Excel's Failure to Launch from the Planning Workbench	287
Glossary	291
Index	293

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support for Hearing-Impaired Customers

Oracle customers have access to electronic support through My Oracle Support or by calling Oracle Support at 1.800.223.1711. Hearing-impaired customers in the U.S. who wish to speak to an Oracle Support representative may use a telecommunications relay service (TRS). Information about the TRS is available at <http://www.fcc.gov/cgb/consumerfacts/trs.html/>, and a list of telephone numbers is available at <http://www.fcc.gov/cgb/dro/trsphonebk.html>. International hearing-impaired customers should use the TRS at +1.605.224.1837. An Oracle Support engineer will respond to technical issues according to the standard service request process.

1

Getting Started

In This Chapter

Integrated Operational Planning Analysis Process.....	15
Integrated Operational Planning Data Collection Process.....	17
Analysis Roles and Workflows	18
Using Integrated Operational Planning.....	22

Integrated Operational Planning Analysis Process

Oracle Integrated Operational Planning, Fusion Edition leverages actual (measured) data and plan (forecast) data. Data can come from many sources, such as ERP (Enterprise Resource Planning), PDM (Product Data Management), and CRM (Customer Relationship Management) systems.

The original source of data is the *execution database*. Data from the execution database is loaded into the *Integrated Operational Planning database*, where it is used in what-if scenarios. Data is updated periodically (for example, nightly or weekly).

In what-if analysis, you change actual and plan values in a set of planning worksheets that together make a planning workbook. You can then evaluate changes in key metrics to compare the impact of plan changes. Key metrics and assumptions vary based on the analysis type selected by the person who created the scenario (the *analysis owner*).

```

graph TD
    PlanChanges[/Plan changes/] --> ReviewApproval{Scenario review or approval}
    ReviewApproval -- "Scenario 'Approved'" --> CommitDB[Commit plan changes to the database]
    ReviewApproval -- "Scenario 'Submitted' if approvers do not approve" --> ReviewFeedback[Review and provide feedback to owner]
    ReviewFeedback --> CreateAlt[Create scenario alternatives]
    CreateAlt --> AddComments[Add comments to scenario and cells]
    AddComments --> SubmitPart[Submit scenario to participants and/or approvers]
    SubmitPart --> ReviewApproval
    ReviewApproval -- "Scenario 'Submitted' if approvers included; otherwise scenario 'Approved'" --> CommitDB
    CommitDB --> IOPDB[(IOP database)]
    IOPDB --> ExceptionsTriggered((Exceptions triggered))
    ExceptionsTriggered --> ViewDetails[View exception details]
    ViewDetails --> ExceptionsHomePage((Exceptions on user's Home page))
    ExceptionsHomePage --> SelectAnalysis[Select analysis type]
    SelectAnalysis --> CreateScenario[Create scenario to resolve exceptions]
    CreateScenario --> SelectParticipants[Select participants and approvers]
    SelectParticipants --> ModifyForecast[Modify actual or forecast plan values; recalculate]
    ModifyForecast --> CompareMetrics[Compare key metrics]
    CompareMetrics --> ReviewExceptions[Review scenario exceptions]
    ReviewExceptions --> ViewReports[View reports and charts]
    ViewReports --> AddComments
    ModifyForecast --> CreateAlt
    CompareMetrics --> PerformWhatIf[Perform what-if analysis]
    
```

The flowchart illustrates the scenario planning process, starting with plan changes and leading to either a completed scenario or one in progress. The process involves reviewing and approving scenarios, committing changes to the database, and handling exceptions. It also includes a feedback loop for scenarios that are not approved, allowing for alternative creation and submission. The process concludes with a what-if analysis, comparison of key metrics, and review of scenario exceptions.

Constraints ensure that actual values do not exceed or lag behind planned values. For example, in Manufacturing Operations, a constraint can be defined for excess inventory to set a limit on the amount of material on hand. If this limit is exceeded, an excess inventory exception is generated.

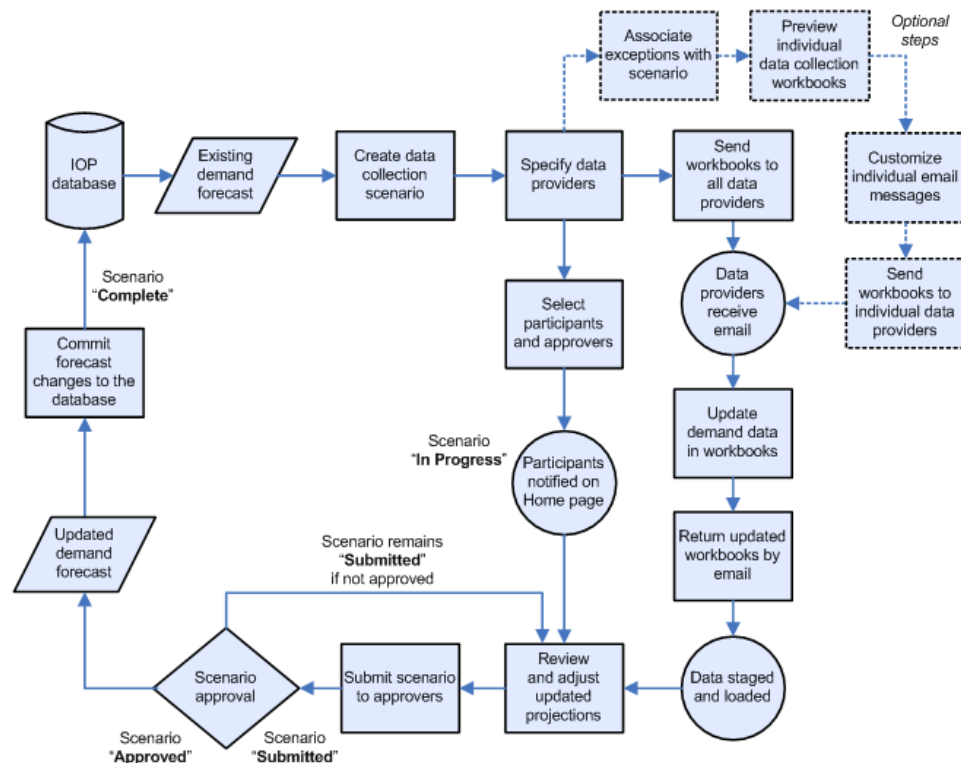
Planners receive notifications on their Home page when exceptions occur. These exceptions can then be resolved through what-if analysis. In [Figure 1](#), a planner can adjust values in the weekly production schedule to eliminate the exception.

Integrated Operational Planning Data Collection Process

The data collection process is similar to the what-if analysis process, with the addition of *data providers* who provide updated forecast data. Updates are consolidated in the database, where analysis owners can examine the impact of forecast changes.

For example, in a demand consolidation scenario, sales representatives provide updated demand data. Usually, data providers are not Integrated Operational Planning users and do not need to log on to the user interface.

Figure 2 Data Collection Process in a Data Collection Scenario



Consider a data collection scenario used for demand consolidation. The existing demand forecast is used as a starting point to create *data collection workbooks*. Each data provider receives a different data collection workbook as an e-mail attachment containing the person's current forecast data. Data providers update the workbooks in Excel and return them by e-mail. When received, the updated workbook data is automatically loaded into Integrated Operational Planning where forecast changes become data changes in the data collection scenario. Analysis owners can generate and send workbooks to all data providers simultaneously, or they can send them individually with customized e-mail messages.

An audit log enables analysis owners to monitor the status of data providers and the data-loading process. If a data provider has not yet returned a workbook, or if a returned workbook contains incorrect data, the analysis owner can resend the workbook. A data collection scenario can be submitted for approval regardless of whether all data providers have successfully responded. Analysis owners can complete data collection workbooks on behalf of a data provider.

Analysis owners can invite participants and approvers to be involved in a data collection scenario, just as for a what-if analysis. In this case, the approver role includes reviewing the forecast numbers input by data providers. The approver can adjust related measure values. Participants can help adjust forecast numbers. For example, participants can adjust different sets of forecast numbers depending on a data provider geographic region.

When a data collection scenario is approved, the forecast numbers are committed to the base data in Integrated Operational Planning, where others can view the adjusted forecast. In the adjusted forecast, there may be two measures for each updated forecast number: one that represents the updated value supplied by data providers, and one that is an adjusted value. For example, if the measure Pipeline Sales represents values supplied by data providers, the measure Projected Sales might be defined as the adjusted value after other considerations are taken into account.

Exceptions can occur in data-collection scenarios. For example, a constraint in a demand consolidation scenario may be defined to flag incomplete or incorrect data. Exceptions can then be used to indicate when action is needed by the analysis owner.

Analysis Roles and Workflows

Subtopics

- [About Analysis Roles and Workflows](#)
- [Analysis Owner](#)
- [Participant](#)
- [Approver](#)
- [Data Provider](#)
- [Administrator](#)

About Analysis Roles and Workflows

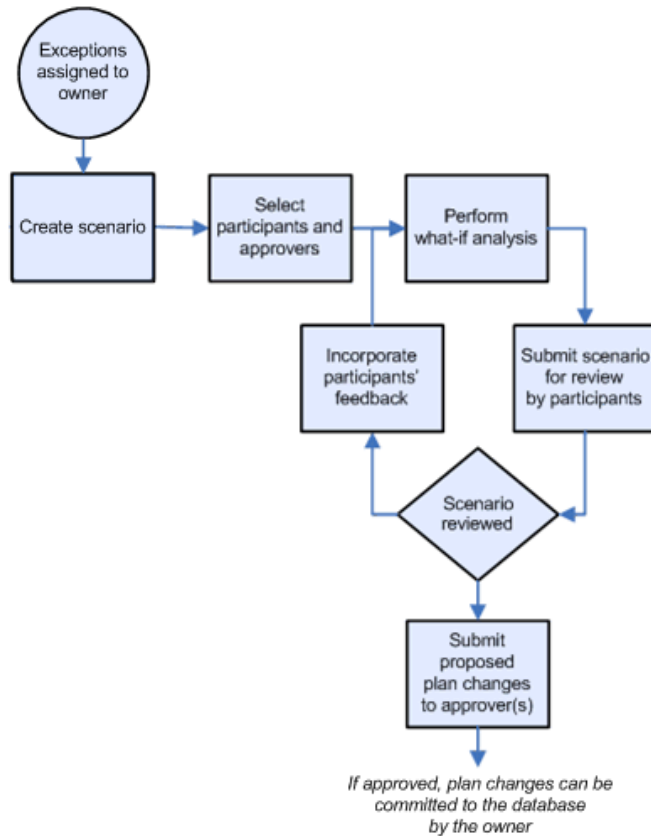
Tasks in Integrated Operational Planning vary based on analysis roles. The tasks that you perform depend on the role during an analysis. You can have one or more roles as you participate in multiple analyses.

The following analysis roles refer to typical workflows in Integrated Operational Planning; they are not formal roles and do not require specific permissions. Variations on these workflows are possible, depending on how you decide to collaborate on an analysis.

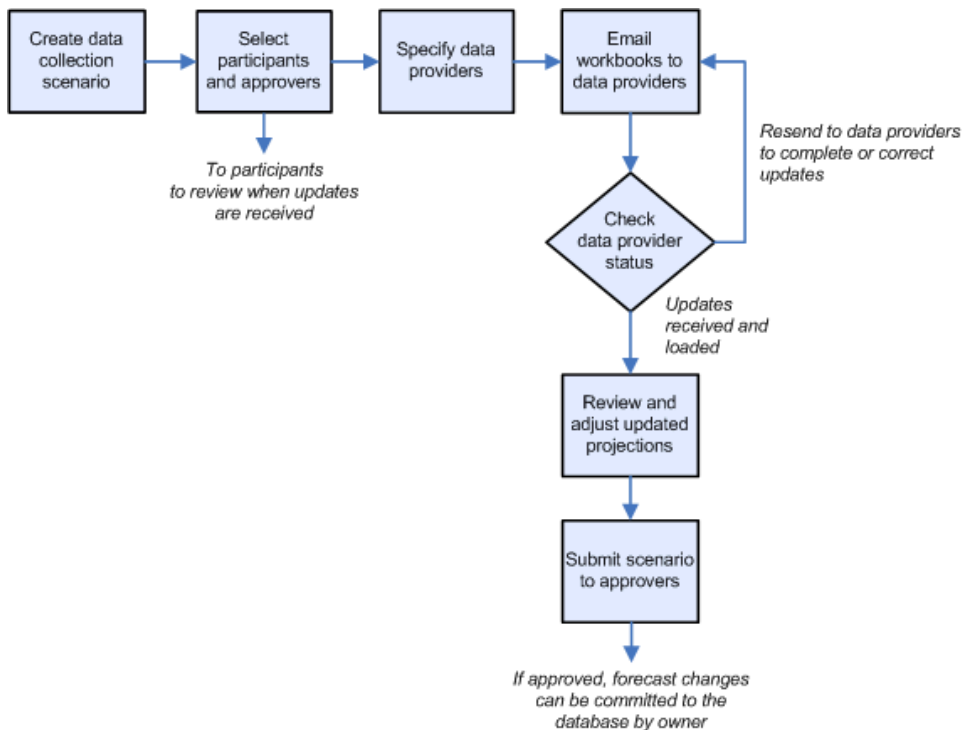
Analysis Owner

Analysis owners create scenarios to perform an analysis. They select analysis types appropriate to the functional area investigated. Analysis owners then create scenarios, invite participants, select approvers, and perform the analysis. When all plan changes are approved, analysis owners commit the changes to the Integrated Operational Planning database.

Figure 3 Typical Workflow for an Analysis Owner



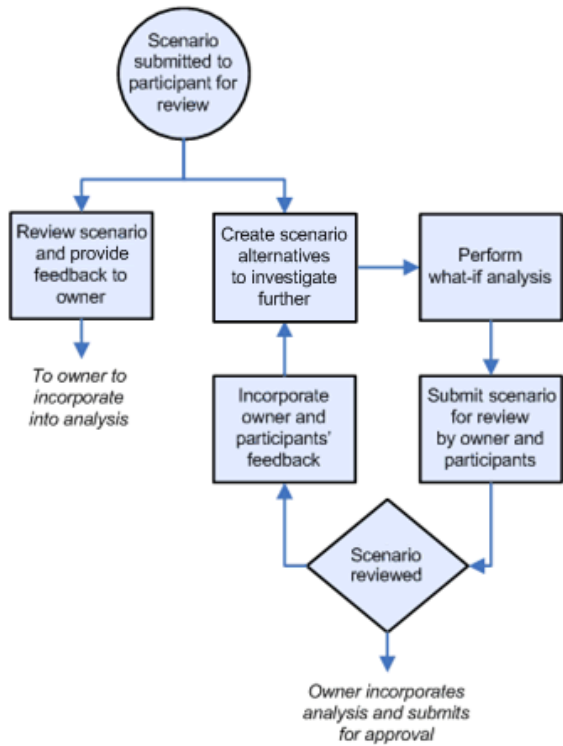
In a data collection scenario designed to gather updated demand forecast data, the typical workflow for an analysis owner includes additional steps:



Participant

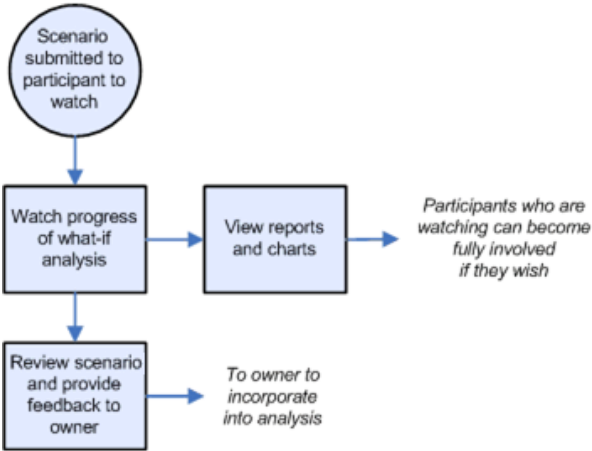
Participants collaborate on an analysis to review scenarios and provide comments to the analysis owner.

Figure 4 Typical Workflow for an Analysis Participant



Some participants primarily intend to track the progress of an analysis. For example, a participant may want to view only reports and charts, or only provide comments to the analysis owner.

Figure 5 Typical Workflow for a Participant Watching an Analysis



Approver

Approvers are given authority to approve proposed plan changes developed as a result of an analysis. If all approvers approve an analysis, the analysis owner can commit plan changes to the Integrated Operational Planning database. If an analysis is not approved, it remains in the Submitted state for further collaboration, or it is deleted.

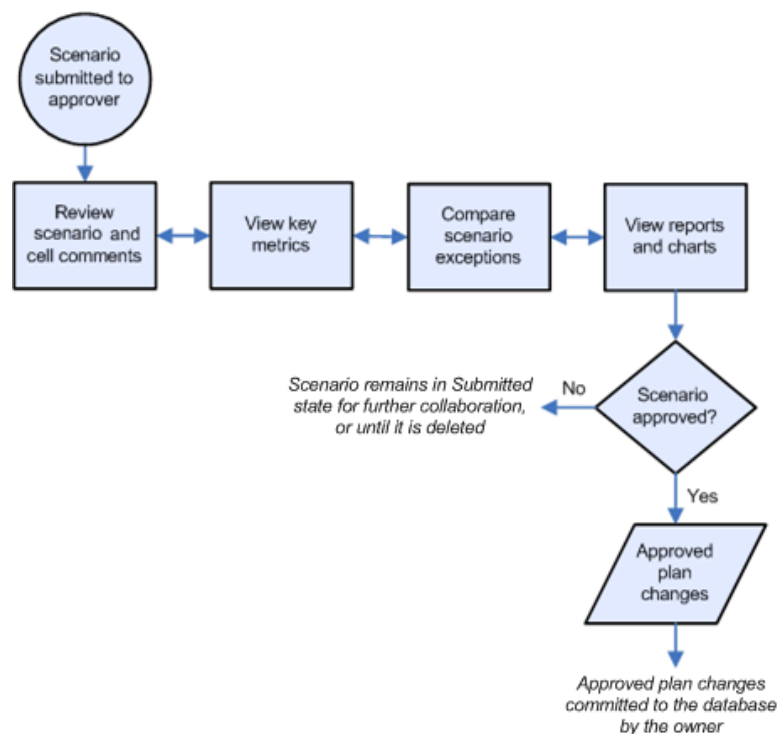
Adding and Deleting Approvers

Approvers defined in analysis type can be added or removed only by an administrator.

Approvers added by the owner of scenarios can be removed only by the owner or administrator.

Participants cannot add or remove approvers.

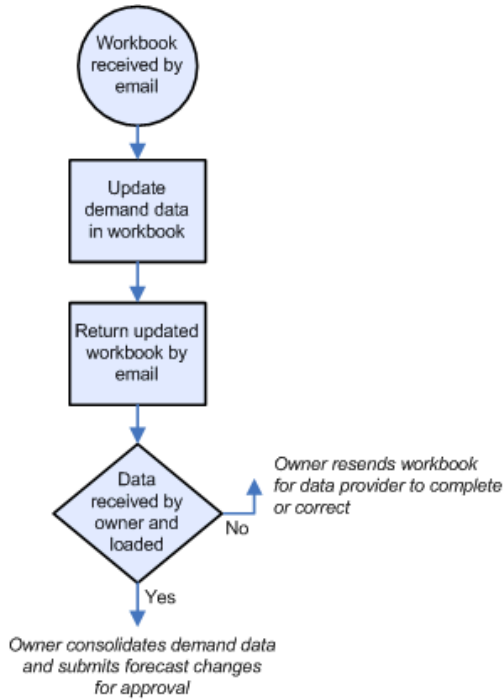
Figure 6 Typical Workflow for an Analysis Approver



Data Provider

Data providers are e-mailed report workbooks and asked to provide updated report data. For example, they may be asked to update demand forecast data which is then consolidated with other demand data in a data collection scenario. Data providers do not directly participate in scenario analysis.

Figure 7 Typical Workflow for a Data Provider



Administrator

Administrators use the Administration Workbench to create the underlying model; view and update the model; and perform tasks such as managing analysis types, worksheets, workbooks, queries, dimensions, and cubes. By default, only administrators can assign new exceptions to users.

See [Chapter 3, “Using the Administration Workbench.”](#)

Using Integrated Operational Planning

Subtopics

- [Logging in to Integrated Operational Planning](#)
- [Understanding the Integrated Operational Planning User Interface](#)

Logging in to Integrated Operational Planning

To log on to Integrated Operational Planning, you must be registered as a user and assigned a user name and password by the administrator.

► To log on to Integrated Operational Planning:

- 1 In an Internet Explorer browser window, enter the following URL:

`http://SERVER_NAME:PORT_NUMBER/interlace`

where *server_name* is the name of the computer where the Integrated Operational Planning server is running, and *PORT_NUMBER* is the HTTP port number set up by the administrator. By default, the port number is 27080. For example:

`http://localhost:27080/interlace`

2 In the Log On page, enter the user name and password and click Log On.

Passwords are case-sensitive.

Understanding the Integrated Operational Planning User Interface

The Integrated Operational Planning user interface provides several ways to work with exceptions, create and monitor scenarios, perform an analysis, view reports and charts, and perform administrative tasks.

The top right of the screen displays a user name and contains a Help link to open the online help and a Logout link to log out of the system.

If you log on as the administrator, you can toggle between the Planning Workbench and the Administration Workbench.

About the Planning Workbench

The Planning Workbench includes the following sections, which you access by selecting a link on the left of the screen.

- [Home](#)—Displays exceptions, scenarios, tasks, and the weekly calendar
- [Analysis Workbench](#)—Used to create and modify scenarios
- [Review Exceptions](#)—Displays the exceptions assigned to you
- [Review Reports](#)—Displays predefined tabular reports and charts
- [Manage Tasks](#)—Displays tasks related to specific analysis types or scenarios
- [Scripts](#)—Displays scripts to perform specific activities in Integrated Operational Planning
- [Statistical Forecasts](#)—Displays statistical forecasts created with ForecastPro or using `isadmin` commands (only available in systems that use forecast collection)
- [User Profile](#)—Manage the first and last name, e-mail address, and select user constraints

The screen title shows the location in the Planning Workbench. The title includes the section, subsection, current task, and the path to navigate to that screen.

Note: See [Chapter 2, “Using the Planning Workbench.”](#)

About the Administration Workbench

The Administration Workbench is visible only if you log with administrator rights. It includes the following sections:

- **Model**—Create and edit data models. Manage data sources, row sources, dimensions, and cubes. See [“Using the Model Tab” on page 78](#).
- **Presentation**—Create and edit business model. Manage analysis types, queries, worksheets, and workbooks. See [“Using the Presentation Tab” on page 111](#).
- **Administration**—Configure database connections, script templates, security filters, and system tools. Manage users and the job queue. See [“Using the Administration Tab” on page 127](#).

2

Using the Planning Workbench

In This Chapter

Planning Workbench Pages	25
Working with Scenarios.....	29
Working with Data Providers	41
Working with Exceptions.....	46
Managing Tasks	47
Reviewing Reports	50
Setting Access to View Reports.....	51
Performing an Analysis	52

Planning Workbench Pages

Subtopics



- [Home](#)
- [Analysis Workbench](#)
- [Review Exceptions](#)
- [Review Reports](#)
- [Manage Tasks](#)
- [Scripts](#)
- [Statistical Forecasts](#)
- [User Profile](#)

Home

The screen title shows where you are in the Planning Workbench. The title includes the section, subsection, current task, and the path to navigate to that screen.

The Home page is displayed after you log on to Integrated Operational Planning. The Home page gives you quick access to the sections discussed in [Table 1](#).

Table 1 Home Page Sections

Section	What Is Displayed
Weekly Calendar	<p>Scenarios and tasks due each week.</p> <p>To change the week, click .</p> <p>To open a scenario or task that is due, click the scenario or task displayed in red.</p> <p>To open a scenario planning workbook, click  next to the scenario.</p>
Exceptions	Exceptions assigned to you.
Scenarios	<p>Scenarios that you are involved as owner, reviewer, or approver, and scenarios that are past due.</p> <p>Scenarios have the following states:</p> <ul style="list-style-type: none"> ● In Progress—Newly created or currently in use in an analysis ● Submitted—Submitted for approval ● Approved—Submitted and approved (You can no longer analyze an approved scenario.) ● Complete—Final plan changes committed to the Integrated Operational Planning database <p>Click a link to review the following types of scenarios:</p> <ul style="list-style-type: none"> ● Overdue—Scenarios that are overdue ● Owner—Scenarios that you own ● Reviewer—Scenarios in which you are a participant ● Approver—Scenarios awaiting approval <p>To create a scenario, click Create Scenario and select an analysis type. You must be assigned at least one analysis type for you to create a scenario.</p>
Tasks	<p>Tasks that you requested or are assigned, or that are past their due date or deferred. Tasks are action items associated with an analysis type or scenario.</p> <p>Tasks have the following states:</p> <ul style="list-style-type: none"> ● Not Started—Newly created by a requester and assigned to a task owner ● In Progress—Currently being acted on by the task owner ● Deferred—On hold ● Waiting on Someone Else—Completion depends on another person ● Completed—Completed as requested <p>Click a link to review the following task types:</p> <ul style="list-style-type: none"> ● Overdue—Overdue tasks you are involved in ● Assigned—Tasks assigned to you as owner ● Requested—Tasks that you requested ● Deferred—Deferred tasks in which that you are involved <p>To create a task, click Create Task and select an analysis type. At least one analysis type must be assigned to you for you to create a task.</p>
Scripts	Scripts to perform specific activities in Integrated Operational Planning

Section	What Is Displayed
Statistical Forecasts	Statistical forecasts created with ForecastPro or using <code>isadmin</code> commands Note: The Statistical Forecasts link is displayed only in systems that use forecast collection.

Analysis Workbench

The Analysis Workbench is where you create and modify scenarios. (See “[Working with Scenarios](#)” on page 29.) It displays the scenarios for each analysis type that the administrator assigned to you. You must be granted access to at least one analysis type to create scenarios.


Each analysis type represents a business area such as forecast consolidation, demand generation, capacity planning, or supply-and-demand alignment. Analysis types are represented by either a planning workbook or a data collection workbook, which are assembled from predefined planning worksheets, where you change plans and investigate the impact of changes on a default set of key metrics and assumptions.

You can perform the following tasks in the Analysis Workbench:

- Select analysis types and create scenarios
- Specify summary information for scenarios, such as name, description, and priority
- Invite users to be participants, approvers, or both
- Define tasks and assign them to users
- Change plan and actual data values in a scenario’s planning workbook and recalculate key metrics, key assumptions, and other measures. (Exceptions are highlighted in red.)
- Compare recalculated key metrics, key assumptions, and the number and type of introduced exceptions to determine the effect of changes

When you click an analysis type in the Analysis Workbench, the sections discussed in [Table 2](#) are displayed.

Table 2 Analysis Workbench Sections

Subsection	Description
Perform Analysis	Create a scenario or modify a scenario for the selected analysis type. Click  to open the planning workbook and begin analysis.
View Exceptions	Displays exceptions specific to the analysis type that you subscribed to.
View Reports	View predefined tabular reports and charts associated with the selected analysis type.
View Tasks	View tasks associated with an analysis type or scenario.

Review Exceptions

The Review Exceptions page displays a list of exceptions assigned to you. Click an exception to view details.

See [“Working with Exceptions” on page 46](#) and [“Creating Constraints” on page 97](#).

Review Reports

The Review Reports page displays predefined tabular reports and charts. The Reports page is similar to the View Reports option in the Analysis Workbench except that it lists all reports available, not just those associated with a particular analysis type.

See [“Review Reports” on page 28](#).

Manage Tasks

The Manage Tasks page displays tasks related to analysis types or scenarios. As a task requester, you enter a text description of the task, set a due date, and assign the task to a task owner.

Task status is displayed on the Home pages of the requester and owner. For tasks associated with a scenario, task status can be changed by the requester, owner, or scenario participants while the scenario is In Progress. For tasks not associated with a scenario, task status can be changed anytime.

For example, you can define tasks that are action items frequently associated with a particular analysis type. You can also use tasks as reminders to follow up on action items.

See [“Managing Tasks” on page 47](#).

Scripts

The Scripts page displays scripts that you can invoke to perform specific activities in Integrated Operational Planning. The scripts displayed are created in the Administration/Script Templates section in the Administration Workbench.

See [“Working with Scripts” on page 71](#) and [“Managing Script Templates” on page 133](#).

Statistical Forecasts

The Statistical Forecasts page displays statistical forecasts created with ForecastPro or using `isadmin` commands. Statistical forecasts are available only in systems that use forecast collection.

See [“Working with Statistical Forecasts” on page 72](#).

User Profile

The user profile includes the full name, e-mail address, and login password.

► To change the user profile:

- 1 Log on to Integrated Operational Planning using the existing user name and password.

- 2 From the **Home** window, click **User Profile** on the navigation panel.

The **Update Profile** window is displayed..

- 3 Change the profile information: First and last name and e-mail address.

Note: Administrators must use the Administration Workbench to change their user profile.

- 4 From the available constraints selected by the Admin, select the constraints that you want to track.

The subset of exceptions shows in the exceptions box on the Home page, the impact window, and in the alerts on the cells.

- 5 Click **OK**.

Working with Scenarios

Subtopics

- [Viewing Scenarios](#)
- [Creating Scenarios](#)
- [Editing Scenarios](#)
- [Undoing Changes](#)
- [Merging Changed Entries](#)
- [Copying Scenarios](#)
- [Comparing Scenarios](#)
- [Deleting Scenarios](#)
- [Submitting Scenarios for Approval](#)
- [Approving Scenarios](#)
- [Completing Scenarios](#)
- [Printing Scenarios](#)
- [E-mailing Scenarios](#)
- [E-mailing an Excel Shortcut to a Scenarios Planning Workbook](#)
- [Adding Scenarios to Favorites](#)

Viewing Scenarios

You can access a list of scenarios from the Integrated Operational Planning home page or from the Analysis Workbench.

Accessing a List of Scenarios From the Home Page

On the Home page, the Scenarios section lists the number of scenarios that you are involved in as owner, reviewer, or approver; and the number of scenarios with past due dates. Click a link corresponding to a scenario type to view a list of scenarios for that type. For example, click Overdue to view overdue scenarios.

Viewing Scenario Details in the Analysis Workbench

Scenario details are displayed in the Analysis Workbench. You can filter the scenarios displayed by selecting an option next to **View**. For example, select Owner to display the scenarios that you are involved in as an owner. To define a custom filter, select Custom Filters next to View.

See [“Using and Customizing Search Filters” on page 45](#).

Basic scenario details include:

- **Scenario**—Scenario name
- **Type**—Scenario type
- **Due Date**—Due date for completing scenario analysis. The due date, set when the scenario is created, is advisory only.
- **Priority**—Priority set when the scenario is created
- **Status**—Current state of the scenario

Scenarios have the following states:

- **In Progress**—Newly created scenarios or ones currently in use in an analysis
- **Submitted**—Scenarios submitted for approval
- **Approved**—Submitted scenarios that are approved. You can no longer analyze an approved scenario.
- **Complete**—Scenarios whose final plan changes are committed to the Integrated Operational Planning database.

The scenario state changes from In Progress to Submitted to Approved. When final plan changes are committed back to the Integrated Operational Planning database, the scenario state changes to Complete. The analysis is an iterative process. An approval flow enables participants and approvers to review an in progress scenario and provide feedback. Each scenario can generate additional exceptions for further investigation.


- **Owner**—Analysis owner who created the scenario. The analysis owner can commit plan changes in a scenario to the Integrated Operational Planning database after the scenario is approved.

Tip: Click a column header to sort the list of scenarios. For example, click Due Date to sort the scenarios from earliest to latest due date. Click Due Date again to reverse the sort order.

Viewing Scenarios for Specific Analysis Types

To view scenarios for a specific analysis type, select an analysis type on the left of the Analysis Workbench. For example, select Demand Impact to display demand impact scenarios.

For each analysis type selected, you can accomplish these tasks:

- **Perform Analysis**—Click a scenario name to view additional details or edit information about the scenario, or click  to open the planning workbook and perform what-if analyses.

- **View Exceptions**—View details for exceptions assigned to you.
- **View Reports**—View predefined tabular reports and charts associated with the selected analysis type.
- **View Tasks**—View tasks associated with an analysis type, create and assign tasks, and update task details.

Note: The analysis types that appear in the Analysis Workbench are defined by an administrator on the Presentation tab in the Administration Workbench.


Creating Scenarios

When you create a scenario, you invite users to become involved as participants or approvers. If you create a data collection scenario, you select data providers to update forecast data. You can create scenarios only with analysis types to which you have been granted access by an administrator.

To create a data collection scenario, you must select a data collection analysis type. Usually, you can recognize a data collection analysis type by its name; for example, Demand Consolidation or MPS Data Import.

► To create a scenario:

- 1 From the **Home** page or from the **Analysis Workbench**, click **Create Scenario** and select an analysis type.
- 2 Enter scenario details:
 - **Name**—Name of the scenario. For example, “2009 Q4 Production Changes.” Scenario names are case-sensitive.
 - **Description**—Description of the scenario. For example, “What-if analysis to address product shortages caused by forecast revisions” or “Data collection to consolidate updated demand data.”
 - **Owner**—Analysis owner. By default, the user who creates the analysis is the analysis owner. Administrators can specify a different analysis owner if desired.
 - **Priority**—Priority of the what-if analysis using this scenario.
 - **Due Date**—Advisory due date for completing scenario analysis.
 - **Participants**—Who can participate in the analysis. (Participants can review comments and provide comments.)

Each analysis type includes a default list of participants. Click  to add or remove participants.


To send a notification message to the selected participants, select **Notify Participants by e-mail**. To send an e-mail message to an individual participant, click the participant’s name.

Invited participants are notified of scenarios to be reviewed in the Scenarios section on their Home page.

- **Approvers**—Who can approve plan changes in the scenario. (Approvers can review and provide comments.)

Each analysis type includes a default list of approvers. Only an administrator can add and remove approvers on the default list.

The analysis owner, participants, and administrators can add additional approvers.

Click  to add additional approvers.

To send a notification to the selected approvers, select **Notify Approvers by e-mail**. To send an e-mail to an individual approver, click the approver's name.

Approvers are notified of scenarios to be approved in the Scenarios section on their Home page.

Note: You can invite a user to be both a participant and an approver. All approvers must approve a scenario before it can be committed to the Integrated Operational Planning database.

- 3 For data collection scenarios, in **Scenario Data Collection**, click the  to add or remove data providers from the list.

Data providers receive data collection workbooks to update forecast data. See [“Working with Data Providers” on page 41](#).

- 4 Do one of the following:

- Click **Save** to save the scenario for later analysis.
- Click **Analyze** to save the scenario and open the scenario planning workbook in Excel.

The scenario state changes to In Progress when it is saved.

Editing Scenarios

Editing a scenario involves changing scenario summary information, assigning new tasks related to the scenario, and changing scenario data collection information

Analysis owners and participants can edit only scenarios that are In Progress. Approvers cannot edit scenarios except to add comments when they approve it.

➤ To edit a scenario:

- 1 In the **Analysis Workbench**, click a scenario name.
- 2 In **Scenario Summary**, edit the desired information.

See [step 2 on page 31](#) for information about the fields in this section.

- 3 In **Comments**, click **Add Comment** to add a note about what was changed.

The comment, the name of the user who added it, and a date/time stamp is saved with the scenario when it is approved and committed to the Integrated Operational Planning database. Participants and approvers can view comments.

4 In Tasks, perform an action:

- Click **Add Task** to add action items associated with the scenario.

See “Viewing Task Details” in [“Managing Tasks” on page 47](#) for information on the task information to enter.

- Click **Associate Task** to associate existing common tasks with the scenario.



The Associate Task button is disabled when there are no common tasks to associate. Common tasks are tasks that have not yet been associated with any scenario.

After you associate common tasks with a scenario, you must save the scenario to persist the association.

You cannot delete a task while editing a scenario. Tasks are deleted in the Tasks section of the navigation panel.

See [“Deleting Tasks” on page 50](#).

5 For data collection scenarios, in **Scenario Data Collection, do one or more of the following:**

- Click  to preview the data collection workbook for each data provider.
- Click  to send a data collection workbook to an individual data provider.
- Click a data provider’s name to send e-mail to the data provider.
- Click **Select Providers** to add or remove data providers from the list.
- Click **Send All** to e-mail a data collection workbook to all data providers selected.
- Click **Show Summary** to display a data provider status summary.

6 In **Key Metrics/Assumptions, review the key metrics and key assumptions assigned to the scenario.**

7 In **Scenario Impact, review information about fixed exceptions, introduced exceptions, scenario changes, and cell comments.**

8 Do one of the following:

- Click **Save** to save the changes for later analysis.
- Click **Actions** and select an action.

Undoing Changes

You can undo changes made to a scenario, but you cannot redo changes.

➤ To undo changes made to a scenario:

- 1 In the **Analysis Workbench**, click a scenario name.**
- 2 Click **Actions** and select **Undo changes**.**

The **Undo Changes** window is displayed.


- 3 To search for changes, enter search values in the grid or select the changes and move them to the **List of changes to discard**.
- 4 Click **OK**.
- 5 To complete the Undo Changes request, click **OK**.

Merging Changed Entries

Merging changed entries involves selecting changes from scenarios to create a new scenario.

► To merge a scenario:

- 1 From the **Home** page or from the **Analysis Workbench**, click a scenario name.
- 2 Click **Actions** and select **Merge Scenarios**.
- 3 Enter scenario details:
 - **Name**—Name of the scenario. For example, “2009 Q4 Production Changes.” Scenario names are case-sensitive.
 - **Description**—Description of the scenario. For example, “What-if analysis to address product shortages caused by forecast revisions” or “Data collection to consolidate updated demand data.”
 - **Owner**—Analysis owner. By default, the user who creates the analysis is the analysis owner. Administrators can specify a different analysis owner if desired.
 - **Priority**—Priority of the what-if analysis using this scenario.
 - **Due Date**—Advisory due date for completing scenario analysis.
 - **Participants**—Who can participate in the analysis. (Participants can review comments and provide comments.)

Each analysis type includes a default list of participants. Click  to add or remove participants.


To send a notification message to the selected participants, select **Notify Participants by e-mail**. To send an e-mail message to an individual participant, click the participant’s name.

Invited participants are notified of scenarios to be reviewed in the Scenarios section on their Home page.

- **Approvers**—Who can approve plan changes in the scenario. (Approvers can review and provide comments.)

Each analysis type includes a default list of approvers. Only an administrator can add and remove approvers on the default list.

The analysis owner, participants, and administrators can add additional approvers.

Click  to add additional approvers.

To send a notification to the selected approvers, select **Notify Approvers by e-mail**. To send an e-mail to an individual approver, click the approver's name.

Approvers are notified of scenarios to be approved in the Scenarios section on their Home page.

Note: You can invite a user to be both a participant and an approver. All approvers must approve a scenario before it can be committed to the Integrated Operational Planning database.

- 4 Select the **Available Change Entries**, add them to the **Selected Change Entries**, and then click **Save**.

Copying Scenarios

Copying a scenario involves copying a source scenario to a target scenario. The source scenario can be in any state.

► To copy a scenario:

- 1 From the **Home** page or from the **Analysis Workbench**, click a scenario name.
- 2 Click **Actions** and select **Copy Scenarios**.
- 3 In **Target Scenario Name**, enter the name of the target scenario.
- 4 In **Copy Options**, select the options that you want copied:
 - **Data Changes**
 - **Cell Comments**
 - **Key Metrics**
 - **Header Information**
- 5 Click **OK**.

Comparing Scenarios

► To compare scenarios:

- 1 From the **Home** page or from the **Analysis Workbench**, click the scenarios that you want to compare.
- 2 Click **Actions** and select **Compare Scenarios**.
- 3 Click the **Key Metrics** tab or the **Key Assumptions** tab or the **Data Changes** tab to view the comparisons.

Deleting Scenarios

► To delete a scenario:

- 1 In the **Analysis Workbench**, click the check box beside the scenarios.

To delete all the scenarios for an analysis type, click **Select All**. (The Select All and Deselect All links appear only if you have fewer than 100 scenarios.)

Note: If you are the analysis owner, you can delete scenarios in the In Progress state, the Complete state, or the Approved state, but not the Submitted state.

- 2 Click **Actions** and select **Delete scenarios**.
- 3 Click **OK** to delete the selected scenarios.

Submitting Scenarios for Approval

Analysis owners and participants can submit scenarios for approval. Only scenarios that are In Progress state can be submitted. All approvers must approve a scenario before it can be committed to the Integrated Operational Planning database.

► To submit a scenario for approval:

- 1 In the **Analysis Workbench**, click a scenario name.
- 2 Click **Actions** and select **Submit Scenario Name for Approval**.

where **Scenario Name** is the name of the selected scenario.

For data collection scenarios, a message box informs you if tasks in the Scenario Data Collection panel are incomplete. Tasks are incomplete in these situations:

- Data requests have not been sent to all data providers
- Data changes have not been received from all data providers
- Data received from data providers has not been successfully loaded

For example, a task may be incomplete if the analysis owner or a participant entered data changes on behalf of a data provider.

Either click **Cancel** and complete the incomplete tasks, or click **OK** to submit the scenario for approval without completing all tasks. (You cannot complete tasks after the scenario is submitted.)

- 3 In **Submit Scenario Name for Approval**, do the following:
 - a. Enter any desired comments for the approver. The comments are saved with the scenario.
 - b. Select whether to send e-mail notifications to the analysis owner, participants, or approvers.

Send notification e-mail to is enabled only when the sender and at least one receiver has a valid e-mail address.

If scenario data is out-of-date relative to the base data in Integrated Operational Planning, you can refresh scenario data with the latest base data and then submit the scenario for approval, or you can refresh scenario data with the latest base data, but do not submit the scenario for approval. Continue the analysis using the updated data.

4 Click **OK**.

The entire scenario moves to the Submitted state, where it remains until all approvers approve it. If no approvers are selected for a scenario, it automatically moves to the Approved state.

Note: In the Scenario Impact panel, a check mark is displayed next to the name of a submitted scenario.

Changing Submitted Scenarios with Deleted Approvers Back to an “In Progress” State

If you submit a scenario and then delete or deactivate a pending approver, you can change the scenario back to an “In Progress” state.

➤ To change submitted scenarios with deleted approvers back to an “In Progress” state:

1 Log on as the administrator or as the owner of the scenario.

2 On the Scenario Summary page, click the following error message:

Pending approver has been deleted from Hyperion Shared Services.
Click here to change the scenario state to edit mode.

3 Change the scenario state.

The scenario owner and approver are notified by e-mail that the scenario has been changed back to edit mode, and the following comment is added in the Scenario Summary page:

Pending approver of submitted scenario has been deleted from
Hyperion Shared Services. Scenario state has been changed to edit
mode.

Removing Deleted Users From Owners, Participants, or Approvers Lists

When a user is deleted or deactivated in Oracle's Hyperion® Shared Services, the user name is displayed in red in the list of users available to assign as owners, participants, or approvers. You must remove the deleted user before proceeding with scenario creation or analysis.

➤ To remove a deleted user assigned as an owner:

1 Log on as the administrator.

2 Remove the deleted owner.

3 Assign a new owner.

➤ To remove a deleted user assigned as a participant or approver:

1 Log on as the administrator or as the owner.

2 Remove the deleted participant or approver.

3 Assign additional participants or approvers as desired.

Approving Scenarios

All approvers must approve a scenario before it can be committed to the Integrated Operational Planning database. Approvers can review the analysis in Excel, add cell comments, and add scenario comments. Approvers cannot change any data values or recalculate measure values.

Before approving a scenario, you can review the scenario.

After a scenario is approved, the planning workbook for the scenario is unavailable for viewing in Microsoft Excel.

► To approve a scenario:

- 1 **Display the scenarios you are involved in as approver.**
 - In the Scenarios section on the Home page, click Approver.
 - In the Analysis Workbench, select Approver in the View menu.
- 2 **Click a scenario name and review scenario information.**
- 3 **In *Approve this scenario?*, Click Yes.**
- 4 **In *Approve Changes*, enter any desired comments, and select whether to send an e-mail notification to the analysis owner, participants, or other approvers when you submit the approval or denial.**
- 5 **Click *Save* to submit approval.**

When all required approvers have submitted their approval, the entire scenario moves to the Approved state. You cannot open a scenario in Excel when it is in the Approved state. When a scenario is approved, the scenario data changes are automatically committed to the base data in Integrated Operational Planning.

Completing Scenarios

Subtopics

- [Committing Plan Changes to the Database](#)
- [Viewing Analysis History](#)

Committing Plan Changes to the Database

When all approvers have approved a scenario, the scenario final plan changes are committed to the base data in Integrated Operational Planning. Key Metrics, Key Assumptions, and Scenario Impact information are saved with the scenario. The analysis owner can then move the scenario from the Approved state to the Complete state.

Note: Individual worksheet cell data changes and cell comments exist only while a scenario is in progress. They are not saved when a scenario is committed to the Integrated Operational Planning database.

- To move a scenario from the Approved state to the Complete state:
 - 1 In the **Analysis Workbench**, click the name of a scenario with a status of **Approved**.
 - 2 In **Apply Changes**, enter comments and select whether to send an e-mail notification to the analysis owner, participants, or approvers when the scenario is complete.
 - 3 Click **Commit**.

The entire scenario moves to the Complete state.

Note: You can view an analysis history for completed scenarios; however, you cannot open them in Excel. Scenarios must be moved to the Complete state before they can be deleted.

Viewing Analysis History

Summary information saved with a completed analysis provides a useful audit trail for later review. When plan changes in a scenario are committed to the Integrated Operational Planning database, the following information is saved and can be viewed later by the analysis owner:

- Scenario summary information, including the list of participants and approvers
 - Scenario comments, including who entered them and when (time and date)
 - Key metrics, key assumptions, and their values for the scenario
- To view a scenario's analysis history, select the scenario in the Analysis Workbench, or click a scenario on the Home page.

The number of days completed scenarios are available for viewing is set by the `ui.analysis.keepAppliedWorkbooksDays` parameter in the `ISServer.properties` file. To access this file, go to:

`INTERLACE_ROOT\interlace\config\ISServer.properties`

where `INTERLACE_ROOT` is the Integrated Operational Planning installation directory.

The default value for this parameter is 30 days.

Printing Scenarios

You can print details for scenarios in any state. Scenario details include:

- Scenario summary
- List of data providers (for data collection scenarios)
- Scenario comments
- Actual key metric values in the Integrated Operational Planning base data and key metric values calculated for the scenario
- Scenario impact, including the number of fixed and introduced exceptions, the record of changes to data values, and any cell comments

- To print a scenario:
 - 1 In the **Analysis Workbench**, click a scenario name.
 - 2 Click **Actions** and select **Print this Page**.
 - 3 Select printer options and click **Print**.

E-mailing Scenarios

You can e-mail details for scenarios in any state. Scenario details include:

- Scenario summary
- List of data providers (for data collection scenarios)
- Scenario comments
- Actual key metric values in the Integrated Operational Planning base data and key metric values calculated for the scenario
- Scenario impact, including the number of fixed and introduced exceptions, the record of changes to data values, and any cell comments

- To e-mail a scenario:
 - 1 In the **Analysis Workbench**, click a scenario name.
 - 2 Click **Actions** and select **E-mail this Page**.
 - 3 In **E-mail this Page**, enter the e-mail addresses of desired recipients (separated by commas), the e-mail subject, and any comments; then, click **Send**.

E-mailing an Excel Shortcut to a Scenarios Planning Workbook

When you e-mail an Excel shortcut to a scenario planning workbook, the shortcut is sent as a URL. You can e-mail shortcuts only for scenarios that are In Progress or Submitted.

- To e-mail an Excel shortcut:
 - 1 In the **Analysis Workbench**, select a scenario in the In Progress or Submitted state.
 - 2 Click **Actions** and select **E-mail Excel Shortcut**.
 - 3 In **E-mail Excel Shortcut**, enter the e-mail addresses of desired recipients (separated by commas), the e-mail subject, and any comments; then, click **Send**.

Adding Scenarios to Favorites

When you add an Excel shortcut to a scenario planning workbook, the shortcut is automatically added to the Internet Explorer Favorites menu.

You can create shortcuts only for scenarios that are In Progress or Submitted. If a scenario is no longer available (if it is approved, complete, or deleted), the shortcut opens the Integrated Operational Planning login screen.

When you select the Excel shortcut from the Favorites menu, the scenario planning workbook opens in Excel.

► To add an Excel shortcut to the Favorites menu:

- 1 In the **Analysis Workbench**, select a scenario in the In Progress or Submitted state.
- 2 Click **Actions** and select **Add Excel Shortcut to Favorites**.
- 3 In the Internet Explorer Add Favorite dialog box, enter a name and select a location for the shortcut.

After adding the shortcut to the Favorites menu, you can save the shortcut on the Windows desktop or e-mail it to another user. You can also e-mail an Excel shortcut from the Integrated Operational Planning Analysis Workbench.

Working with Data Providers

Subtopics

- [About Data Providers](#)
- [About Data Collection Scenarios](#)
- [Assigning Data Providers to a Data Collection Scenario](#)
- [Sending Workbooks to Data Providers](#)
- [Monitoring Data Provider Status](#)
- [Using and Customizing Search Filters](#)

About Data Providers

Data providers receive data collection workbooks in which they update and return forecast data. Updates are consolidated in the database, where analysis owners can examine the impact of forecast changes. For example, in a demand consolidation scenario, sales representatives provide updated demand data.

Usually, data providers are not Integrated Operational Planning users and do not need to log on to the Planning Workbench. If they do log on, they cannot view scenarios or forecast data, unless they are also the analysis owner, a participant, or an approver.

About Data Collection Scenarios

Data collection scenarios are similar to a scenarios for what-if analysis, with the addition of data providers. Two workbooks are used in data collection scenarios:

- **Planning Workbook**—Used by the analysis owner, participants, and approvers for what-if analysis to examine the impact of forecast changes sent by data providers. Just as for nondata

collection scenarios, you can refresh scenario data, change and recalculate measure values, and review scenario impact.


- **Data Collection Workbook**—Used by data providers for manual entry of updated forecast data. A data collection workbook consists of at least one report worksheet containing current forecast data obtained from the base data in Integrated Operational Planning. Each data provider receives a different version of the workbook that is automatically filtered to display their forecast data.

A data collection workbook includes a hidden load specification used by Integrated Operational Planning to stage and load updated data. Unlike planning workbooks, data collection workbooks cannot be used for what-if analysis. They do not display the Integrated Operational Planning menu, toolbar, or context menus. Data collection workbooks are for manual data entry and updates only.

Note: See [“Integrated Operational Planning Data Collection Process”](#) on page 17.

Assigning Data Providers to a Data Collection Scenario

➤ To assign data providers to a new data collection scenario:

- 1 In the **Analysis Workbench**, click **Create Scenario** and enter information about the scenario.
- 2 In **Scenario Data Collection**, click  to add or remove data providers.


➤ To assign data providers to an existing data collection scenario:

- 1 In the **Analysis Workbench**, select a scenario.
- 2 In **Scenario Data Collection**, click **Select Providers** to add or remove data providers.

Note: Administrators must create an Integrated Operational Planning user for each data provider before the data provider can be added to a data collection scenario.

Sending Workbooks to Data Providers

➤ To send data collection workbooks to data providers:

- 1 In the **Analysis Workbench**, click a data collection scenario name.
- 2 In the **Scenario Data Collection** section, perform an action:
 - Click  to send a data collection workbook to an individual data provider.
 - Click **Send All** to send a data collection workbook to all data providers.

Each data provider receives a different version of the data collection workbook containing the data provider’s current forecast data. Data providers do not see each other forecast data unless the provider is also the analysis owner, a participant, or an approver for the scenario.

3 In the **Send Data Collection Form** dialog box, select the template to use for the e-mail sent to data providers and enter the requested information:

- **Base Template**—Default template for the e-mail, including information about the data providers, cc list, subject, and message. With the exception of data providers (which are read-only), you can add or edit the information and click **Send**. When you click Send, you are asked whether to save the template as a user template.
- **User Template**—Last saved user template. If no user template is saved, the default base template is displayed.

When data providers receive a data collection workbook from an analysis owner, they should:

1. Open the data collection workbook in Excel.
2. Review and update workbook data using the existing format for data entries.
3. Save changes to the workbook file in Excel.
4. (Optional) Contact the analysis owner by replying to the original e-mail message.
5. Return the updated workbook (.xls file) as an e-mail attachment.

When an updated workbook is received, the Integrated Operational Planning server adds a set of data staging and loading jobs to the job queue.

Note: You can preview and send data collection workbooks only for scenarios in the In Progress state. The workbook file name is automatically generated with the file extension .xls. The file name is different for each data provider.

Monitoring Data Provider Status

After sending data collection workbooks, you can monitor the status of data collection tasks to confirm that data providers receive, update, and return their workbooks and that data is loaded successfully. Analysis owners and participants can view status when the scenario is in any state. Approvers can view status when the scenario is in the Submitted state.

A hidden worksheet in each workbook contains a reference to a load specification file on the Integrated Operational Planning server. The load specification instructs the server how to load the incoming data. When received, the updated workbook file (.xls) is saved to the server in the `INTERLACE_ROOT\custom\data` directory, where `INTERLACE_ROOT` is the Integrated Operational Planning installation directory.

Data is prepared for loading in a staging table and is loaded into the Integrated Operational Planning database. The server immediately adds a set of data staging and loading jobs to the job queue with a Pending status. A separate set of jobs is added for each data provider. Jobs are executed by a batch script that is scheduled to run periodically. An administrator can also run jobs manually.


Analysis owners and participants review and adjust the forecast as needed before submitting the scenario for approval. The analysis owner can make corrections on behalf of the data provider and resend the corrected workbook to the server as an e-mail attachment. The analysis owner

then commits the approved scenario with the updated forecast to the base data in Integrated Operational Planning.

► To monitor data provider status:

1 In the **Analysis Workbench**, select a scenario.

2 In **Scenario Data Collection**, review the **Status**:

- **No Action**—The data collection workbook was not sent to the data provider.
- **'Send E-mail' Pending**—The data collection workbook was sent, and the process is pending.
- **'Received E-mail' Succeeded**—The server has received an updated workbook from the data provider. Click  to open the workbook and view updated forecast data from the data provider.
- **'Stage Data' Pending**—The server is adding a set of data staging and loading jobs to the job queue. You can wait for a scheduled batch script to execute the jobs, or an administrator can execute jobs manually in the Administration Workbench

Data staging and loading jobs are in the demand job category (also called job type). If the administrator selects a job in the demand category to run manually, pending demand jobs for data providers in the job queue are executed in order.

The status may change to Issued in the Data Provider Status window during execution of long jobs. For short jobs, the Issued status is displayed only in the console log.

- **Stage Data Succeeded with Warnings or Stage Data Failed**—The staging job was completed with warnings or interrupted with errors. Validation checks are performed before staging to ensure that data has the expected data type. If warnings or errors occur, data content may be the source. The analysis owner should contact the data provider to correct the data and resend the workbook.
- **Load Data Succeeded with Warnings or Load Data Failed**—Data loaded into the scenario sandbox was completed with warnings or interrupted with errors. Detailed validation is performed before data is loaded. An error log is generated, and an information icon is displayed next to the Load Data job in the Data Provider Status window. Click the information icon to open a second window with error details.

Tip: For all statuses except No Action, you can click the status to open a dialog box that displays details about data provider status.

Using and Customizing Search Filters

Subtopics

- [Filtering a List](#)
- [Adding a Custom Search Filter](#)
- [Editing a Custom Search Filter](#)
- [Deleting a Custom Search Filter](#)

Filtering a List

You can filter scenarios, exceptions, tasks, or statistical forecasts to view those that you own, those in a particular state, or those that you requested.

- To filter a list of scenarios, exceptions, tasks, or statistical forecasts, select a filter from the View menu, at the top of the Analysis Workbench, the Exceptions screen, the Tasks screen, or the Statistical Forecasts screen.

The filters that are available vary for scenarios, exceptions, tasks, and statistical forecasts.

Adding a Custom Search Filter

- To add a custom search filter:

- 1 From the **View** menu, select **Custom Filters**.
- 2 In **Custom Filters**, click **Add**.
- 3 In **Add Custom Filter**, enter a filter name and add filter restrictions.

For example, depending on the filter type, you can add restrictions for State, Priority, Description, Name, Status, and Subject.

- 4 Click **OK** to keep the selections; then, click **OK** again to save the custom filter.

Editing a Custom Search Filter

- To edit a custom search filter:

- 1 From the **View** menu, select **Custom Filters**.
- 2 In **Custom Filters**, select a filter and click **Edit**.
- 3 In **Edit Custom Filter**, modify the filter.
- 4 Click **OK** to keep the changes; and then click **OK** again to save the filter.

Deleting a Custom Search Filter

► To delete a custom search filter:

- 1 From the **View** menu, select **Custom Filters**.
- 2 In **Custom Filters**, select a filter and click **Remove**.
- 3 Click **OK** to delete the selected filter.

Working with Exceptions

Subtopics

- [About Exceptions](#)
- [Reviewing Exceptions](#)

About Exceptions

Exceptions are events triggered in Integrated Operational Planning when configured business rules or thresholds are violated.

Types of exceptions:

- **Fixed Exceptions**—Occur when base data violates a constraint, when updated data is loaded from the execution database, or when approved data changes in a scenario are committed to the database. Fixed exceptions are resolved during analysis.
- **Introduced Exceptions**—Introduced during analysis when data changes in a scenario violate a constraint

Exceptions have the following states:

- **Assigned**—Assigned to you by another user
- **Closed**—Fixed in an approved scenario

The exception state changes from Assigned to Closed as analysis proceeds.

Reviewing Exceptions

► To review a list of exceptions:

- 1 In the **Planning Workbench**, perform an action:
 - Go to the Review Exceptions screen.
 - On the Home page, under Exceptions, select an exception name.
- 2 To sort the list, select a column header.
- 3 From the **View** menu, select a different search filter, or select **Custom Filters** to add a custom filter.

4 To view addition details, highlight an exception.

Exception details:

- **Name**—Exception name
- **ID**—Unique number for each exception
- **Owner**—User currently assigned the exception
- **Priority**—Set when the exception is first defined or when the exception is assigned to a user
- **Due Date**—Date for closing (resolving) the exception. Set when the exception is assigned to a user and is advisory only.
- **Status**—Current state of the exception
- **Scenario**—Scenario connected with the exception
- **Description**—Exception description

Managing Tasks

Subtopics

- [Viewing Tasks](#)
- [Creating Tasks](#)
- [Editing Tasks](#)
- [Deleting Tasks](#)

Viewing Tasks

Tasks are action items associated with an analysis. A task can be part of a scenario, or it can stand alone. You can use tasks to track or follow up on actions that are performed frequently during analysis. The user who requests the action item assigns it to a task owner to complete.

➤ To view tasks:

1 Access a list of tasks and their details from the following areas:

- **Home Page**—Tasks section lists the number of tasks you are involved in that are overdue, assigned, requested, or deferred. Click a link corresponding to a task type to view a list of tasks for that type. For example, select **Deferred** to view all deferred tasks.
- **Navigation Panel**—To access a list of tasks from the navigation panel, select **Manage Tasks** on the left side of the Planning Workbench. You can filter the tasks displayed by selecting an option next to **View**. For example, select **Overdue** to display overdue tasks. (To define a custom filter, select Custom Filters next to view.

See [“Using and Customizing Search Filters” on page 45](#).

- **Analysis Workbench**—Tasks in the Analysis Workbench are associated with scenarios. To view the tasks associated with a scenario, select the scenario and review the tasks in the Tasks section.

2 Select a task name to view or edit the complete list of details:

- **Subject**—Task description
- **Due Date**—Task due date (Past due dates are displayed in red on the Home pages of the requestor and task owner.)
- **Start Date**—When the task should start
- **Status**—Not Started, In Progress, Deferred, Waiting On Someone Else, or Completed
For newly created tasks, select Not Started. For tasks associated with scenarios, the requestor, task owner, and scenario participants can change the task status while the scenario is In Progress.
- **Priority**—Low, Normal, or High
- **% Complete**—Percentage of the task that is complete
- **Requester**—User who created the task
- **Owner**—User who is assigned the task. The task name is displayed on the Home page of the task owner.
- **Scenario**—Scenario associated with the task, if any
- **Comments**—Additional information about the task and its progress
- **Send notification e-mail to**—Notifies the task requestor, owner, or scenario participants when the task is saved

Note: Only requesters, owners, and administrators can view stand-alone tasks. Requesters, owners, scenario participants, and administrators can view tasks associated with a scenario.

Creating Tasks

When you create a task, you give it a name, description, and due date. As the task requester, you then assign the task to yourself or another user to complete.

Any user can create a stand-alone task, even if that user has no assigned analysis types. To create a task associated with a scenario, you must be the scenario owner, a participant, or an administrator. You can create tasks associated only with scenarios in the In Progress state.

➤ To create a task:

1 Do one of the following:

- From the Home page, in the Tasks section, click Create Task and select an analysis type.
- In the navigation panel, click Manage Tasks; and then click Create Task.

- In the Analysis Workbench, select an analysis type and click View Tasks; then click Create Task. (You can use this method only if you have assigned analysis types or if you are a scenario participant.)
- While creating a new scenario or editing an In Progress scenario, go to the Tasks section and click Add Task.

2 Enter task details.

See “Viewing Task Details” in [“Managing Tasks” on page 47](#) for information on the task information to enter.

3 Click **Create**.

If you created the task from within a scenario, click OK to return to scenario details screen, and then click **Save**.

Note: The analysis types displayed when you create a task are defined by an administrator on the Presentation tab in the Administration Workbench.

Editing Tasks

Requesters, owners, and administrators can edit stand-alone tasks in any state.

Requesters, owners, scenario participants, and administrators can edit tasks associated with scenarios in the In Progress state.

Requesters, owners, and administrators can edit tasks associated with scenarios in the Submitted, Approved, or Complete state; however, they must edit task information in the Tasks section in the navigation panel. Approvers cannot edit tasks.

➤ To edit a task:

1 Do one of the following:

- From the Home page, click a link under Tasks to select a task type.
- In the navigation panel, select Manage Tasks and select a task type from the View menu.
- In the Analysis Workbench, select an analysis type and click View Tasks.
- While editing an In Progress scenario, go to the Tasks section and select the name of a task.

2 Edit task details.

See “Viewing Task Details” in [“Managing Tasks” on page 47](#).

3 Depending on how you accessed the Tasks screen, click **Save** or **OK**.

Deleting Tasks

- To delete a task, click the check box next to the task and click **Delete Tasks**.
- To delete all the tasks for an analysis type, click **Select All**. (The Select All and Deselect All links appear only if you have fewer than 100 tasks.)

Note: You cannot delete a task while editing a scenario.

Reviewing Reports

Use the reports in Integrated Operational Planning to track changes in key metrics and assumptions and to monitor the occurrence of certain exceptions. For example, you can use an Impact report to view the revenue impact of plan changes over time; you could use a Forecast report to track changes in plan revenue; and you can use a Shortage report to review parts and product shortages that led to broken constraints.

You can review reports by using the base data in Integrated Operational Planning or by using the data in a scenario, and you can refresh reports to display the latest base data or scenario data.

If a report depends on base data, it includes approved plan changes committed to the database. A report that depends on a scenario includes plan changes in an analysis that may be in progress. Scenario data is separate from the base data until plan changes are committed.

- To review a report:

- 1 In the navigation panel, click **Review Reports** and select a report.

Available reports depend on the configuration.

- 2 If prompted, select whether to run the report against base data in Integrated Operational Planning or against a scenario.

Some reports can be run only against base data or only against scenario data. In these cases, you will not be prompted to select an option.

- 3 If you are running the report against a scenario, select a scenario and click **OK**.

You can run reports only against scenarios in the In Progress or Submitted state. If the window is empty, no scenarios exist in these states.

Excel opens and displays the report.

- 4 **Optional.** Select filters to narrow the range of displayed data.

If no filters are defined, the report includes all data ranges; for example, data for all products, all locations, or all time periods.

- 5 Click **Refresh Data**.

Note: Some reports enable you to submit changes to the database. The submission process may take time to complete. Do not close Excel until the submission process is complete.

Note: Some planning workbooks include report worksheets in addition to planning worksheets. Report worksheets display predefined tabular reports or charts.

Setting Access to View Reports

You can set access so that users can only see one report in an Integrated Operational Planning model. For example, if users need to see a report in a model to which they do not have access, you can configure access to view the report in the model.

➤ To configure access to a particular report in a model:

- 1 **Create an analysis type.**
- 2 **Associate the report to the new analysis type.**
- 3 **Add the user to this analysis type.**

The user then inherits the ownership (or visibility) through the analysis type.

Note: See [“Creating Analysis Types” on page 112](#).

Performing an Analysis

Subtopics

- [Opening a Scenario in Excel](#)
- [Viewing Dimensional Data](#)
- [Changing and Recalculating Measure Values](#)
- [Reviewing the Impact of Plan Changes](#)
- [Adding Cell Comments](#)
- [Viewing a Scenario as Another Analysis Type](#)
- [Refreshing Scenario Data](#)
- [Working with Scripts](#)
- [Working with Statistical Forecasts](#)

Opening a Scenario in Excel


Subtopics

- [Understanding Planning Workbooks](#)
- [Changing the Worksheet Layout](#)

After creating a scenario in Integrated Operational Planning, use the analysis tools in Excel to modify data values in the planning workbook, recalculate key metrics and assumptions, and view scenario impact details.

Excel is used to display different views of the modeling data in the Integrated Operational Planning database. The Integrated Operational Planning server performs analysis calculations.

➤ To open a scenario planning workbook in Excel:

- 1 In the **Integrated Operational Planning Analysis Workbench**, display scenarios.
- 2 Do one of the following:
 - Click  to the left of a scenario name.
 - Click the scenario, then click **Actions**, and then select *Analyze Scenario Name*.

You can also click the Excel shortcut for the scenario from the Favorites menu in Internet Explorer. See [“Adding Scenarios to Favorites” on page 40](#).

Note: Analysis Owners and Participants can analyze scenarios in Excel when they are in the In Progress state. Approvers can review scenarios in Excel when they are in the Submitted state. You cannot open scenarios in the Approved or Complete state in Excel.

Tip: If Excel takes a long time to open or does not launch, see [“Excel-Related Issues” on page 286](#).

Understanding Planning Workbooks

A *planning workbook* is a predefined Excel workbook used in what-if analysis. Each planning workbook is associated with an analysis type and is designed to examine the effect of plan changes on a set of key metrics and assumptions.

A planning workbook is assembled from *planning worksheets*. A planning worksheet provides a view of modeling data in the Integrated Operational Planning database. Each planning worksheet displays a subset of the modeling data and can be included in more than one planning workbook, so its data can be used by different analysis types.

A planning workbook may also include *report worksheets*. A report worksheet is a predefined Excel worksheet that uses tabular reports and charts to display data in the Integrated Operational Planning database. The data displayed is defined using queries.

You navigate through a planning workbook as you would in any Excel workbook. The name of the active worksheet is displayed at the top left of the worksheet and on the bottom sheet tab. Switch worksheets by selecting the sheet tabs.

You can modify the following elements of a planning worksheet:

- [Worksheet Layout](#)
- [Worksheet Dimensions](#)
- [Worksheet Measures](#)
- [Worksheet Cells](#)

Worksheet Layout

Layouts define a view into the underlying data. The layout defines which dimension members and measures are displayed. You define whether the layout is used for the current scenario or for all scenarios with a specific analysis type and whether the layout is used as the initial view.

Worksheet layouts are user-specific, private, sheet-specific, and analysis type-specific. You can create layouts for different working areas and store initial layouts for different sheets of the same analysis type.

Tip: Users working in different geographies and with different products can bookmark their data area as their initial layout.

See [“Changing the Worksheet Layout”](#) on page 56.

Worksheet Dimensions

Columns and rows in planning worksheets represent categories of business data called *dimensions*. Common dimensions in planning worksheets include Geography, Product, Customer, and Time. Dimensions provide indexes to specify the location of each data cell in a worksheet.

A dimension contains individual elements called *members*. For example, the Time dimension includes members that correspond to each week, month, quarter, or year.

The members of a dimension can be organized into *hierarchies* to represent parent-child relationships. For example, the Time dimension can have a hierarchy named Fiscal to represent the fiscal calendar and another hierarchy named Manufacturing to represent the manufacturing calendar.

A dimension hierarchy has *hierarchy levels* that represent levels in the hierarchy from top to bottom. For example, the Time dimension includes the hierarchy levels Year, Quarter, Month, and Week. The dimension members at the Quarter level are specific, individual quarters such as Q1 '04, Q2 '04, Q3 '04, and Q4 '04.

Note: The dimension and hierarchy names in planning workbooks depend on how the worksheets are defined. Names can be customized.

In a planning worksheet, the Time dimension is generally represented by a row. Other dimensions, such as Geography, Product, and Customer, are represented by columns.

Worksheet Measures

Measures, the names of numerical data values in planning worksheets, are members of a special dimension, called the *measure dimension*. Examples of measures are Projected Sales, Booked Units, and Actual Average Selling Price.

Generally, measures are displayed in the rightmost dimension column, to the left of the vertical line that separates dimension columns from the data values in the worksheet. You can expand and collapse the column of measures, just as you can for other dimensions.

Measures in a planning worksheet depend on how the worksheet is defined for an analysis type. Each analysis type has predefined *key metrics*, which are measures of special interest for that analysis type. For example, the Supply Impact analysis type has the key metrics Excess Material Costs, Expedite Costs, and Scrap Metal Costs.

In an analysis, you make plan changes by modifying data values in a worksheet. Then you recalculate all data values, which include key metrics, to view the impact of the changes.

Measure values can be detailed or summarized, depending on whether dimension hierarchies in the worksheet are expanded or collapsed. When dimension hierarchies are collapsed, summarized measure values are displayed. When dimensions are expanded, more-detailed breakdowns of measure values are displayed.

For example, if the Time dimension for the measure Shipped Units is fully collapsed, the number of Shipped Units for all time is displayed. If the Time dimension is expanded by one level, the number of Shipped Units for each year is also displayed. If the Time dimension is fully expanded, the number of Shipped Units is detailed for each year, quarter, month, and week, depending on the worksheet.

Tip: You can customize worksheets, including the layout of dimensions and measures, text formatting, and cell highlighting. For example, you can highlight the current time period in the Time dimension in a different color to easily locate the current year, month, day, or week.

Worksheet Cells

Each measure value in a worksheet cell is located at an intersection of dimensions. If a dimension hierarchy is collapsed, then corresponding cells contain summarized measure values. If a dimension hierarchy is expanded, cells contain more-detailed measure values. When you review scenario impact, the locations of cells containing values for key metrics are displayed.

A *cell location* can be expressed as a combination of one member from each dimension, including the measure dimension. For example:

NA x All Customers x Fiscal/2005 Q2 x BDF x Booked Sales

The previous expression gives the cell location for the number of booked sales of product family BDF to all customers in North America in the second quarter of fiscal year 2005. In this expression:

- Geography dimension = NA (North America)
- Customer dimension = All Customers
- Time dimension = Fiscal/2005 Q2 (the second quarter of fiscal year 2005)
- Product dimension = BDF (a product family)
- Measure = Booked Sales

The number of booked sales is a more-summarized value. If the Geography, Customer, Time, and Product dimensions are expanded further, more-detailed values for booked sales can be displayed. For example:

West x OEM x FY 2005/July x A234 x Booked Sales

where

- Geography dimension = West (a territory in North America)
- Customer dimension = OEM (OEM customers)
- Time dimension = FY 2005/July (the month of July in fiscal year 2005)
- Product dimension = A234 (a model number in product family BDF)
- Measure = Booked Sales

When an exception occurs because of a constraint violation, the corresponding worksheet cells are highlighted in red. For example, if measure values for inventories of certain finished goods exceed thresholds for the week, corresponding worksheet cells are highlighted. A text comment is added to highlighted cells to describe the exception

You can add comments to individual cells by right-clicking the cell to communicate with other participants and to review their comments. Cells with comments are displayed with a red triangle

in the upper right corner. When you move the cursor over a highlighted cell or one with a red triangle, the cell comment is displayed.

Some measures have predefined *in-line reports* that provide transaction-level or component-level details for measure values. For example, the measure Booked Sales might have an in-line report called Orders that displays sales order data for a measure value. The in-line reports available depend on how the worksheets are defined.

Changing the Worksheet Layout

You can change display options for a planning worksheet in Excel using:

- [Oracle Toolbar and Menu Commands](#)
- [Context Menus](#)


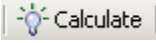




Oracle Toolbar and Menu Commands






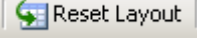

The Oracle toolbar and menu are displayed when you open a planning workbook that contains planning worksheets.

The toolbar and menu are not displayed if you open a report workbook that contains only report worksheets, or in data collection workbooks sent to data providers in a data collection scenario.

➤ To access the Oracle toolbar and menu, on a planning workbook, click the **Add-Ins** tab.

Table 3 Oracle Toolbar and Menu Commands

Toolbar Button	Menu Command	Description
	Show Impact Window	Opens the Impact window for the current analysis type and scenario. See “Reviewing the Impact of Plan Changes” on page 68 .
	Calculate Data Changes	Recalculates key metrics, key assumptions, and measure values in a planning workbook. Exceptions introduced as a result of a data change are highlighted in red. After recalculating, you can review scenario impact. This option is displayed for the analysis owner and participants when a scenario is In Progress or Submitted.
	Update	Refreshes scenario data with the latest base data in Integrated Operational Planning This option is displayed for approvers when a scenario is in the Submitted state.
	Summary	Displays the Scenario Summary dialog box, where you can review information about the scenario you are currently analyzing in Excel.
	Scripts	Displays the scripts defined in Integrated Operational Planning. Click a script name to execute a script.
	Reports	Displays the reports defined in Integrated Operational Planning. Click a report name to run the report.

Toolbar Button	Menu Command	Description
 Focus	Focus on Selection	Focuses on the selected dimension members and measures so that only corresponding data is displayed in the worksheet
	N/A	Searches for a dimension member or measure, even if it is not currently displayed in the worksheet (for example, when a hierarchy is collapsed). See “Searching Dimensions and Measures” on page 61 .
 Save Layout	Save Layout	Saves the current planning worksheet layout
 Select Layout	Select Layout	<p>Selects from user-defined planning worksheet layouts</p> <p>You can see both your layout and the administrator layout. If the selected layout is invalid, the administrator layout is displayed. If the administrator layout is also invalid, summary members for all dimensions are displayed. Layouts can become invalid if a member in the layout is no longer active in the system. For example, if it is a new year, a member in the old year that is referenced in the layout could become inactive.</p> <p>The initial worksheet layout is identified by a <i>_template</i> suffix. For example, <i>Sales Forecast_template</i>.</p>
N/A	Delete Layout	Deletes a planning worksheet layout
 Previous Layout	N/A	<p>Returns to the previous worksheet layout</p> <p>Integrated Operational Planning stores up to six previous layouts.</p>
 Reset Layout	Reset to Default Layout	Resets the planning worksheet layout to the original view, before changes were made
 View Display Options	N/A	<p>Defines how search results are displayed:</p> <ul style="list-style-type: none"> ● How should search results be displayed? Options: Show next level down, Show next level up, Show both, Show bottom, and Show results only ● What should be displayed? Options: Show Excel formulas, and Hide empty rows ● Where to show subtotal? Options: Show on the top and Show at the bottom ● How control level zoom navigation be performed? Options: Show level members within member's hierarchy (Default) and Show all level members See “Showing Hierarchy Levels” on page 61. ● How should reconciliation notification be performed? Select: Prompt after every data change
N/A	Apply Styles and Formatting	<p>Defines new styles and text formatting for cells in a planning worksheet</p> <p>You can change the style and text formatting that applies to dimensions, measure values, exceptions, in-line reports, or planning worksheet headers.</p>
N/A	Manage Measure Sets	<p>Defines a set of measures to view in the worksheet</p> <p>The name of the measure set is added to the context menu. To change which measures are displayed, select Show Summary or Show Detail.</p>
N/A	Control Level Zoom	Displays or hides a dimension's Show Summary and Show Hierarchy Level menu items

Toolbar Button	Menu Command	Description
N/A	Manage In-line Reports	Adds, edits, or deletes in-line reports You select the source of data for the report, define whether users can manually update the report, and select which measure values in the planning worksheet enables you to display the report.
N/A	View Display Options	Selects search options to use when searching for dimension members or measures. Shows measure formulas and hides empty rows
N/A	Oracle IOP Help	Opens the Integrated Operational Planning online help

Context Menus

Context menus are displayed in planning workbooks that contain planning worksheets used in a what-if analysis. The context menus are not displayed in data collection workbooks sent to data providers in a data collection scenario.

- To view a context menu, right-click a worksheet cell containing a hierarchy level name, measure name, or measure value.

The context menu that opens depends on the type of cell selected.

Table 4 Context Menu Commands

Menu Command	Description
Zoom In	Expands the current branch of a dimension hierarchy by one level
Zoom Out	Collapses the current branch of a dimension hierarchy by one level
Include Parent	Expands the current branch of a dimension hierarchy one level up. (Only available for static, sparse dimensions that exclude time and measure dimensions.)
Include Subtotal	Calculates and displays a subtotal for the members selected. The members do not have to be hierarchically related. (Available only for static, sparse dimensions that exclude time and measure dimensions.) To include a subtotal, ensure that Show Excel Formula option under the View Display Options menu is enabled; otherwise, the cells will not show values.
Show Summary	Collapses the dimension hierarchy to show the top level. Collapses the list of measures to show a subset of the measures in the worksheet. You can customize this set of measures.
Show Hierarchy Level	Expands or collapses the dimension hierarchy to show the specified hierarchy level
Show Detail	Expands the list of measures to show a larger set of the measures in the worksheet. You can customize this set of measures.
Show Debug Measures	Expands all measures in the cube, including hidden ones (available only to administrators)
Show Measure Set Name	Expands or collapses the list of measures to show the specified set of measures. You can define measure sets and add or remove measures from a set. Tip: Create measure sets that are logical collections such as Inventory Details, Demand Breakdown, Various Forecasts, Calculation Breakdown, or Actual Data.

Menu Command	Description
Select Measures	Opens a dialog box where you can modify the measures currently displayed (only available for the measure dimension)
Show Description	Displays a description for the dimension member or measure name, if available
Show Formula	Displays the measure formula
Insert/Edit Comment	Enters a cell comment for a measure value. Commented cells are indicated by a red triangle in the upper right corner of the cell.
Add to Key Metrics	Adds a measure value to the list of key metrics for the scenario. Added values apply only to the current planning workbook while the scenario is In Progress. You can also add a measure value permanently to the list of key metrics for a given analysis type. Values for key metrics are calculated and displayed when you view scenario impact details.
In-line Report Name	Displays an in-line report below a measure value giving transaction-level details for that value. In-line reports may not be available for all measure values. Some in-line reports can be updated manually.
Clear Report	Clears the displayed in-line report for a measure value

Viewing Dimensional Data

Subtopics

- [Expanding and Collapsing Dimensions and Measures](#)
- [Searching Dimensions and Measures](#)

When you open a planning worksheet, dimension hierarchies and measures may be expanded or collapsed to different levels. For example, in the Time dimension, you may see data values for Years and Quarters, but not for Months or Weeks. You may see a single measure or several measures. The initial view depends on how the worksheets are defined.

Expanding and Collapsing Dimensions and Measures

Subtopics

- [Expanding Dimensions](#)
- [Collapsing Dimensions](#)
- [Showing Hierarchy Levels](#)
- [Expanding and Collapsing Lists of Measures](#)

In a planning worksheet, use toolbars and menus to expand (zoom in) and collapse (zoom out) dimension hierarchies and lists of measures to view data values at the level needed for analysis.

See [“Changing the Worksheet Layout” on page 56](#).

Expanding Dimensions

➤ To expand a dimension to view lower levels of the hierarchy, perform an action:

- Right-click a dimension member and select Zoom In.

The branch of the hierarchy directly below the selected dimension member is expanded. You can zoom in until the branch is fully expanded.

- Right-click a dimension member and select Show *Hierarchy Level*, where *Hierarchy Level* is a lower level in the hierarchy.

The hierarchy expands to show that level. For example, right-clicking on a Geography dimension may show options such as Show Region and Show Territory.

Tip: To achieve the best system responsiveness, expand dimension hierarchies only to the level needed for the current analysis.

Collapsing Dimensions

➤ To collapse a dimension to view higher levels of the hierarchy, perform an action:

- Right-click a dimension member and select Zoom Out.

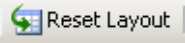
The hierarchy collapses so that only the parent of the selected dimension member is displayed. You can zoom out until the branch containing the selected dimension member is fully collapsed.

- Right-click a dimension member and select Show *Hierarchy Level*, where *Hierarchy Level* is a higher level in the hierarchy.

The dimension hierarchy collapses to show that level. For example, if a Time dimension is fully expanded, right-clicking on the Month hierarchy level may show options such as Show Quarter and Show Year.

- Right-click a dimension member and select Show Summary.


The hierarchy for that dimension collapses to the top-level view.

- To return to the default view of the dimension hierarchy, click  or select **Oracle**, and then **Reset to Default Layout**.

Note: When a dimension, such as Time, is expanded horizontally in a planning worksheet, the hierarchy moves left to right from lower levels (more detailed, Months) to higher levels (more summarized, Years). The most summarized level (All Time) is at the far right. When a dimension, such as Geography, is expanded vertically in a worksheet, the hierarchy moves top to bottom from lower levels (more detailed, such as Cities) to higher levels (more summarized, such as Countries). The most summarized level (All Locations) is at the bottom. These summation directions are consistent with how Excel shows summations at the bottom of a column of numbers or at the right end of a row of numbers.

Showing Hierarchy Levels

- To set the behavior of the Show Hierarchy level menu:

- 1 In an Excel planning worksheet, click  or select **Oracle**, and then **View Display Options** to open **Options**.
- 2 In the **How does control level zoom navigation perform?** section, select one of the following:
 - To navigate to level members in current member's hierarchy only, select **Show level members within member's hierarchy**. For example, in Fiscal dimension, if you click **Show Month Level** menu on member 2007 (year level), all months for 2007 are listed.
 - To navigate to all level members in a selected level, select **Show all level members**. For example, in Fiscal dimension, if you click **Show Month Level** menu on member 2007 (year level), all months for all years are listed.

Expanding and Collapsing Lists of Measures

- To expand a list of measures, right-click a measure and select **Show Detail**.
- To collapse a list of measures, right-click a measure and select **Show Summary**.

Searching Dimensions and Measures

You can search for a dimension member or measure to quickly display data values scrolled out of view in a worksheet or hidden in a collapsed hierarchy.

You can display data for a particular dimension member or measure. For example, you can display data values for Week 2 May 2004, OEM customers, or Booked Sales.

You can search only one worksheet at a time. Search results include measure values for the search item specified. You can change search options to reduce the amount of data displayed or to display more detail.

- To search for a dimension member or measure:

- 1 On the **Integrated Operational Planning** toolbar, enter a search term in the **Search** field and press **Enter**.

Search terms are case-insensitive, and you can enter all or part of a term.

If the search returns one match, the corresponding data is displayed immediately in the worksheet. If the search returns multiple matches, the **Search Results** window opens to display all results.

For example, to display data only for FY 2004, enter 2004 in the **Search** field and press **Enter**. The **Search Results** window opens to display all possible results that include the term 2004, such as:

Fiscal/Fiscal/2004 Q1

Fiscal/Fiscal/2004 Q2

Fiscal/Fiscal/2004 Q3

Fiscal/Fiscal/2004 Q4

Fiscal/Fiscal/FY 2004

The previous expressions give the fully qualified member names, including the dimension name (Fiscal), namespace (Fiscal), and member names (2004 Q1, 2004 Q2,...). In this example, the names of the dimension and namespace are the same.

Dimension, hierarchy, and member names depend on how the planning worksheets are defined. You can customize names.

2 In the Search Results window, select the items to display by enabling their check boxes.

3 Select how to display search results.

- **Show Next Level Down**—Show data for the search item and for the hierarchy level below it. For example, if the search item is FY 2004, show data for FY 2004 and for Q1 2004 through Q4 2004.
- **Show Next Level Up**—Show data for the search item and for the hierarchy level above it. For example, if the search item is FY 2004, show data for FY 2004 and a summation for All Time.
- **Show Next Level Up and Down**—Show data for the search item and for the hierarchy levels above and below it. For example, if the search item is FY 2004, show data for FY 2004, for Q1 2004 through Q4 2004, and a summation for All Time.
- **Show Bottom**—Show data for the bottom hierarchy level. For example, show data for FY 2004.
- **Show Results Only**—Show data only for the search item. For example, show data only for FY 2004.

To define the default search option for the planning workbook, click



4 **Optional.** Search again using a different search term by entering the search term in **Current search criteria** and clicking **Search Again**.

For example, to change the search to FY 2009, enter 2009 in the Search Results window and click Search Again.

5 Click **OK** to display the data in the worksheet.

Tip: Click the right side of the search area to display and select previous search terms.

Changing and Recalculating Measure Values

Subtopics

- [Selecting the Measures to Display](#)
- [Changing Measure Values](#)
- [Recalculating Measure Values](#)
- [Adding Measures to Key Metrics](#)
- [Working with Custom Measure Sets](#)
- [Using an Excel Formula to Recalculate or Display Measure Value Changes](#)
- [Working with In-line Reports](#)

A what-if analysis involves changing measure values in a planning worksheet and recalculating key metrics, key assumptions, and other measures for the workbook. Analysis owners and participants can change measure values in a scenario's planning workbook when the scenario is in the In Progress state. Approvers cannot change data.

Measure values are loaded or input into the Integrated Operational Planning database, and therefore into planning worksheets, at a particular level in each dimension hierarchy. Values at higher levels in a hierarchy are generally obtained by summation. The level at which values are loaded is the leaf (lowest) level. For example, if monthly values are loaded, the leaf level in the Time dimension hierarchy is Months. Yearly values are then obtained by summation.

After changing a measure value, you recalculate key metrics, key assumptions, and other measures in the planning workbook. The worksheet display is updated, and you drill up or down in the data to view recalculated values. If a data change violates a predefined constraint and introduces an exception, the worksheet cell where the exception occurs is highlighted in red. Data changes in one worksheet can affect other worksheets. For example, a data change in one worksheet can introduce an exception in another worksheet.

During analysis, the analysis owner and participants can add text comments to worksheet cells to inform other participants about data changes and to obtain feedback. A red triangle in the upper right corner of a worksheet cell indicates comments.

Some measures include descriptions or details on how the measure value is calculated. You can also view in-line reports for some measures that show transaction-level or component-level details; for example, purchase order details.

Selecting the Measures to Display

➤ To select the measures to display:

- 1 In an Excel planning worksheet, right-click a measure dimension and choose **Select Measures**.
- 2 In **Edit Measures**, select the desired measures by moving them from **Available Measures** to **Assigned Measures**.
- 3 Click **OK**.

Changing Measure Values

When a measure appears in multiple worksheets, you must change the loaded or input values for the measure, not the calculated values derived from them. Otherwise, the changes are not propagated to loaded or input values for the same measure in other worksheets, and calculation problems may occur. Similarly, when you change a measure value that is summarized at higher levels in a dimension hierarchy, you must make the changes to the leaf (lowest) level in each hierarchy.

For example, assume that values for the measure Projected Sales are loaded based on fiscal calendar months. When you change values for Projected Sales, you must change the monthly values. If you change yearly values, the changes will not be propagated to lower levels in the hierarchy, and calculation problems may occur.

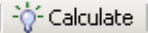
In most cases, before you begin an analysis, you will know with which measures in which worksheets to work. An administrator can determine which measure values are loaded, input, or calculated in a worksheet by looking at measure definition XML files.

- To change a measure value in a planning worksheet, select the cell containing the value, edit the value, and press Enter.



Recalculating Measure Values


When you change data in a planning worksheet, you can update the display to see the recalculated measure values.

- To update the data displayed in a planning worksheet, perform an action:

- Click  **Calculate** or select Oracle, and then Calculate Data Changes.

The toolbar option and menu command are displayed only for the analysis owner and participants, not for approvers.

- Click , or select **Oracle**, and then **Show Impact Window**.
- Click , or select **Oracle**, and then **Focus on Selection**.
- Switch to a different worksheet in the planning workbook.
- Expand or collapse the display of data in the worksheet to view more detailed or summarized measure values.

Tip: Click  **Calculate** to permanently save the recalculated values (as opposed to only updating the display).

Adding Measures to Key Metrics

You can add any measure value in a worksheet to a key metric. For example, you can add the value of a measure during a time period or for a product component to a key metric. The key

metrics that you modify in a planning worksheet are calculated and displayed along with the default key metrics associated with a scenario.

➤ To add a measure to a key metric:

- 1 Right-click the worksheet cell containing the desired measure value.
- 2 Select **Add to Key Metrics**.
- 3 Enter key metric information:
 - **Metric Name**—Identifies the key metric.
 - **Delta**—How to display the key metric value:
 - **Relative**—The difference between the absolute value and the value calculated using the base data in Integrated Operational Planning. For example, the value might correspond to the difference between an actual value and a plan value.
 - **Absolute**—The absolute value of the measure calculated using data in the scenario sandbox. For example, the value might correspond to an actual value entered into a planning workbook during an analysis.
 - **Add to Analysis Type**—Whether to permanently add the key metric to the analysis type associated with the scenario. If you select this option, the key metric will be calculated and displayed for future scenarios created with this analysis type. If you do not select this option, the key metric is associated only with the scenario in this planning worksheet.

Working with Custom Measure Sets

A custom measure set is a collection of measures created for an analysis. For example, you could create a model with two measures sets: one for inventory-related measures and one for forecast-related measures. Measure sets can be associated with dimensions, and they can be level dependent. Each worksheet can have different measure sets.

Creating a Custom Measure Set

➤ To create a custom measure set:

- 1 In an Excel planning worksheet, select **Oracle**, and then **Manage Measure Sets**.
- 2 Click **Add** and enter a measure set name.
- 3 Move measures from **Available Measures** to **Assigned Measures**.
- 4 Click **OK**.

Associating a Measure Set with a Dimension

➤ To associate a measure set with a dimension:

- 1 In an Excel planning worksheet, select **Oracle**, and then **Manage Measure Sets**.

- 2 Select a measure set and click **Edit**.
- 3 Select a dimension and click **OK**.
- 4 Click **OK** again to save the information.

Level-Dependent Measure Sets

In a level-dependent measure set, you associate a measure set with a dimension. Then, for each measure in the measure set, you specify which dimension level the measure applies to, and, therefore, should be displayed.

For example, assume that you have a *Product* hierarchy representing the bill-of-materials for a car. At the top level is a member called *car*. If you zoom in to *car*, you see *wheels*, *doors*, *windshield*, and so on. In the same model is a measure called *MSRP*, the car's listing price, and a measure called *Component Unit Price*, the price for a part. In a measure set that does not contain levels, the hierarchy would appear as shown in [Figure 8](#).

Figure 8 Product Hierarchy That Does Not Have a Level-Dependent Measure Set

Product	Measures	2008 Q1
.. wheels	MSRP	
	Component Unit Price	\$2,200
.. windshield	MSRP	
	Component Unit Price	\$110
.. door	MSRP	
	Component Unit Price	\$120
car	MSRP	23,456
	Component Unit Price	

In a level-dependent measure set, however, the hierarchy would appear as shown in [Figure 9](#).

Figure 9 Product Hierarchy in a Level-Dependent Measure Set

Product	Measures	2008 Q1
.. wheels	Component Unit Price	\$2,200
.. windshield	Component Unit Price	\$110
.. door	Component Unit Price	\$120
car	MSRP	23,456

To create the view in [Figure 9](#), you create a measure set and associate it with the *Product* dimension. In the measure set, you define a *Component Unit Price* measure that should be displayed only for members at the Component level and an *MSRP* measure that should be displayed only for members at the Finished Good level.

► To select measure set levels:

- 1 In an Excel planning worksheet, select **Oracle**, and then **Manage Measure Sets**.
- 2 Select a measure set and click **Edit**.
- 3 Select a dimension and click **OK**.
- 4 In **Assigned Measures**, select a measure.
- 5 Click **Select Levels**, select the desired levels, and click **OK**.

6 Click **OK** again to save the information.

Using an Excel Formula to Recalculate or Display Measure Value Changes

You can use an Excel formula to recalculate or display measure value changes. In addition to the cell value, the Excel formula shows the measure dependency so that you can see how the measure is calculated.

For example, assume you have the following formula:

Ending Inventory = Beginning Inventory + Plan Build

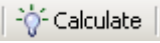
When Show Excel Formula is turned on, for the Ending Inventory measure, in addition to the value, you can see how the measure is calculated. For example:

E5=E4+E3


where E4 is the cell location for the Beginning Inventory and E3 is the cell location for the Build Plan.

If any dependent measures are not in the display, the cell location is replaced by a value. In the previous example, if Beginning Inventory is not in the display (hence the cell address is not available) the formula for Ending Inventory would be:

E5 = 120 + E3

Tip: Click  to permanently save the changes to the dependency measure.

➤ To show Excel formulas:

- 1 In an Excel planning worksheet, click , or select Oracle, and then **View Display Options**.
- 2 Select **Show Excel Formulas**.
- 3 Click **OK**.

Working with In-line Reports

In-line reports are available for measures that provide transaction-level or component-level details. For example, in-line reports for a Forecast worksheet may provide order details (quantity, revenue, and target ship date) and pipeline information (product line, customer name, and expected close date). in-line reports for a Supply worksheet may provide details on purchase order status (vendor and expected receipt date) and part availability (lead time, excess thresholds, and shortage thresholds).

Some in-line reports can be updated, enabling planners to modify data within the report to support what-if analysis. For example, an in-line report with purchase order details enables planners to edit the expected receipt date to model an expedite action. If the expedite action resolves a part shortage, a planner may decide to take further action by confirming with the procurement organization whether the purchase order can be expedited.

The in-line reports in your planning workbooks depend on your worksheets definitions.

Opening In-line Reports

➤ To open an in-line report:

- 1 Right-click a worksheet cell containing a measure value.

If an in-line report is available, its name appears below Add to Key Metrics.

- 2 Select **In-line Report Name**, where *Report Name* is the name of the in-line report.

An in-line report is displayed beneath the cell containing the measure value.


Editing In-line Reports

If an in-line report is editable, it contains fields with values that can be changed. After editing a value, you recalculate data values for the worksheet. If the in-line report field that you changed is not editable, its value reverts to the original value.

If a blank row is displayed at the bottom of an in-line report, you can enter values; then when you recalculate, the new values are inserted into the report.

After recalculating worksheet data values, you can view the effect of the change; for example, you can see whether an exception is resolved.

➤ To edit an in-line report:

- 1 Select a field in the in-line report, change the data value, and press Enter.
- 2 If the in-line report includes a blank row at the bottom, enter values for fields in the row.
- 3 Click  Calculate, or select **Oracle**, and then **Calculate Data Changes**.

The calculate button and menu item are displayed only for the analysis owner and participants when a scenario is in the In Progress or Submitted state.

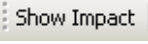
- 4 Review scenario impact to see the impact of your change.

Closing In-line Reports

➤ To close an in-line report, right-click the cell containing the measure value, and then select Clear Report.


Reviewing the Impact of Plan Changes

Use the Impact window to view the results of data changes in a scenario planning workbook.

- To access the Impact window, click , or select **Oracle**, and then **Show Impact Window**.

The following information is displayed:

- **Key Metrics**—Key metrics associated with an analysis type and scenario. Click a value to go to that cell in the worksheet. If a measure value appears on more than one worksheet, select a worksheet destination.
- **Key Assumptions**—Key assumptions associated with an analysis type and scenario. Click a value open a report sheet containing key assumption data.
- **Impact Details**

If the details are calculating, a set of spinning arrows  displays next to Fixed Exceptions, Introduced Exceptions, and Baseline Exceptions.

The system refreshes the window every 30 seconds. The interval is defined in `ISServer.properties` as

```
spreadsheet.impact.refresh.interval=30000
```

Modify the `.properties` file to meet specific requirements. Oracle recommends that you keep this value at a minimum of 30 seconds.

- **Fixed Exceptions**—Number of exceptions that are resolved due to data changes in the planning workbook. Click Show to view cell locations, and then click a location to go to that cell.
- **Introduced Exceptions**—Number of exceptions introduced by a data change in the planning workbook. Click Show to view cell locations, and then click a location to go to that cell.
- **Baseline Exceptions**—All the exceptions in Integrated Operational Planning up to 50.
- **Cell Comments**—Number of cells in the workbook with comments. Click Show to view comments, cell locations, user names, and time stamps for the comments, and then click a location to go to that cell.
- **Data Changes**—Number of data changes made in the planning workbook. Click Show to view data changes, cell locations, user names, and time stamps for the changes, then click a location to go to that cell.

Adding Cell Comments

The analysis owner, participants, and approvers can add comments to data cells in a planning worksheet. Comments can be viewed while a scenario is in the In Progress, Submitted, or Approved state. Cell comments are not saved when a scenario is committed to the Integrated Operational Planning database.

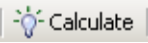
Commented cells are indicated by a red triangle in the upper right corner. Move the cursor over a commented cell to display the comment and the name of the user who inserted or last edited the comment.


► To add cell comments:

- 1 Right-click a cell and select **Insert/Edit Comment**.
- 2 Enter text for the comment.
- 3 Click any worksheet cell to close the comment dialog box.

A red triangle is displayed in the upper right corner of the cell.

- 4 Save the comment.

If you are an analysis owner or participant, click , or select Oracle, and then Calculate Data Changes.

If you are an approver, click , or select Oracle, and then Update.

Viewing a Scenario as Another Analysis Type

You can temporarily switch analysis types to view scenario data from another perspective. When you switch analysis types, you can analyze scenario data in a different planning workbook with planning worksheets, report worksheets, and key metrics and assumptions specific to the temporary analysis type.

For example, if you are analyzing a scenario from a sales perspective using the Demand Impact analysis type, you can switch to a financial perspective that uses the Financial Impact analysis type. You can then calculate key metrics and assumptions and view reports and charts for that analysis type.

When you switch views, data changes and calculated key metrics and assumptions in the temporary view are not saved and do not affect scenario data in the original view. You can switch views for scenarios that are in the In Progress and Submitted states.

► To view a scenario as another analysis type:

- 1 In the Planning Workbench, click **Analysis Workbench**, and then select a scenario.
- 2 Scroll to **Key Metrics/Assumptions**, select an analysis type next to **View as**.

The analysis types that appear in the Analysis Workbench are defined by an administrator on the Presentation tab in the Administration Workbench.

The panel displays key metrics and assumptions for that analysis type.

- 3 Analyze the scenario using the temporary analysis type by clicking **Actions** and selecting **Analyze Scenario Name**, where *Scenario Name* is the name of the scenario.

You are returned to the list of scenarios, and Excel opens to display scenario data in the planning workbook for the temporary analysis type. It may take a few minutes for the scenario to open in Excel.

You can perform what-if analysis in this planning workbook; however, your data changes and calculated key metrics and assumptions are not saved.

- 4 After you complete your analysis using the temporary analysis type, close Excel.

The next time you open the scenario in Integrated Operational Planning, you are returned to the original analysis type with the same scenario data that you had before switching views.

Refreshing Scenario Data

Because the analysis owner and participants can access scenario data simultaneously, data in a planning workbook can become out of date relative to the scenario data. If scenario data is modified by another participant, data in your workbook is updated with the latest scenario data when you switch worksheets or recalculate measure values.

Scenario data also can become out of date relative to the base data in Integrated Operational Planning; for example, when data changes from another scenario are committed to the Integrated Operational Planning database, and if updates to the base data are loaded from your execution database.

When you recalculate measure values, review scenario impact, or submit a scenario for approval, scenario data is checked to determine whether it is out of date. If it is, a warning is displayed. Select an option:

- **Refresh to See Latest Base Data**—Data values in the workbook are updated.
- **Don't Refresh**—Data values in the workbook are updated when you recalculate values. You can continue working with the existing scenario data.

Data changes do not affect the base data in Integrated Operational Planning until the scenario is fully approved.

Tip: Administrators can schedule batch scripts to run periodically to check if scenario data for In Progress scenarios is out of date relative to base data. Scenario data is then updated, and any necessary calculations are performed.

Working with Scripts

Understanding Scripts

Scripts are a set of commands that you can invoke to perform specific activities in Integrated Operational Planning.

For example, you could create a script to import and publish a worksheet, a workbook, and an analysis type as follows:

```
alter model schema set current edit

import worksheet definition from file "xxxxxxxx.xml"
import workbook definition from file "xxxxxxxx.xml"
import analysistype definition from file "xxxxxxxxxxxxxx.xml"

publish metadata changes
```

Scripts are created by an administrator on the Administration or Script Templates page in the Administration Workbench. See [“Adding Script Templates” on page 133](#).

Viewing Scripts

Scripts are displayed in the following areas in the Planning Workbench:

- Scripts section on the Home page
Scripts are displayed on the Home page only if a user has access rights to the script template.
- Scripts section in the navigation panel

Executing Scripts

➤ To execute a script:

- 1 In the Planning Workbench navigation panel, select **Scripts**.
- 2 Select a script and click **Execute**.

Working with Statistical Forecasts

Subtopics

- [Reviewing Forecasts](#)
- [Managing Rules](#)
- [Managing Overrides](#)

About Statistical Forecasts

Statistical Forecasts display forecast data generated with ForecastPro. The Integrated Operational Planning user interface does not provide a way to generate forecast data; however, you can generate forecast data using `isadmin` commands executed through a command-line interface. Forecast data for various durations (daily, weekly, monthly, quarterly, yearly) can be generated by grouping `isadmin` commands into appropriate scripts.

➤ To display the Statistical Forecasts link in the Integrated Operational Planning Planning Workbench, add the following line to `site.properties` and restart the Integrated Operational Planning server.

```
# Control activation of left bar navigation options
navigation.forecast.enabled=true
```

Note: If ForecastPro is installed on the same server as the forecast generator, in `site.properties`, set `forecastpro.isInstalled=true`.

Reviewing Forecasts

Subtopics

- [Viewing Forecasts](#)
- [Analyzing Forecasts](#)

Viewing Forecasts

Forecast data is displayed in the Planning Workbench in a tabular format. Each row contains forecast data that corresponds to a “forecast item.”

For example, in a model with Product, Region, Sales Count, and Time dimensions, you could generate forecast data for each Product-Region combination by considering Sales Count figures at specific time intervals. Each Product-Region combination would form a forecast item.

Forecast data includes information such as the forecast value, the applied algorithm, and the applied rule for each forecast item.

➤ To view a list of forecasts:

- 1 In the Planning Workbench navigation panel, select **Statistical Forecasts**.
- 2 Select **Review Forecasts**.
- 3 **(Optional):** Select a column header to sort the list.
- 4 **Optional:** From the **View** menu, select a different search filter, or select **Custom Filters** to add a custom filter.

Analyzing Forecasts

➤ To view additional forecast details, select a forecast.

Additional details include:

- **Forecast Graph**—Plots historical and forecast data against time. The graph includes a lower and upper bound for each forecast value.
- **Forecast Details**—Algorithm used for forecasting along with additional forecast characteristics such as standard deviation, standard forecast error, mean average, and upper or lower bounds.

➤ To modify the algorithm applied to the forecast:

- 1 Display the Forecast Graph and click **Select Algorithm**.
- 2 Select an algorithm and enter model parameters.
- 3 Click **OK** and review the changes to the Forecast Graph.
- 4 Keep this algorithm or select a different one.
- 5 Click **Accept Algorithm**.

The forecast item moves from the Review Forecasts page to the Manage Overrides page.

Managing Rules

Subtopics

- [Creating Rules](#)
- [Editing Rules](#)
- [Assigning and Applying Sequences to Rules](#)

Note: A new or updated rule is applied to forecast data when the daily, weekly, monthly, quarterly, or yearly script is executed.

Creating Rules

► To create a rule:



- 1 In the Planning Workbench navigation panel, select **Statistical Forecasts**.
- 2 Select **Manage Rules**.
- 3 Click **Add**.
- 4 Enter the following rule details:
 - **Name**—Rule name
 - **Description**—Rule description
- 5 Click **OK**.
- 6 Enter additional rule details:
 - **Action**—Algorithm to apply to the data items in the rule
 - **Criteria**—Conditions for the data items in the rule; for example, Product equals CD Player or Region equals Asia-Pacific
- 7 Click **OK**.

Editing Rules

► To edit a rule:

- 1 In the Planning Workbench navigation panel, select **Statistical Forecasts**.
- 2 Select **Manage Rules**.
- 3 Select the name of a rule.
- 4 Edit details.
- 5 Click **OK**.

Assigning and Applying Sequences to Rules

- To assign and apply a sequence to a rule:
- 1 In the Planning Workbench navigation panel, select **Statistical Forecasts**.
 - 2 Select **Manage Rules**.
 - 3 In **Applied Rules**, move a rule from **Available Rules** to **Assigned Rules**.
 - 4 Click   to define the sequence in which the rules will be applied.
 - 5 Click **Save**.

Note: When a rule is assigned, it is activated. Active rules are applied in the sequence defined. If rules conflict, the active rule highest in the sequence is used.

Managing Overrides

The Manage Overrides page displays the forecast items to which a different algorithm has been applied. After daily, weekly, monthly, quarterly, or yearly scripts are executed, the new algorithms are applied, and the forecast item is displayed again on the Review Forecasts page.

3

Using the Administration Workbench

In This Chapter

About the Administration Workbench	77
Using the Model Tab	78
Using the Presentation Tab	111
Using the Administration Tab	127

About the Administration Workbench

The Administration Workbench includes the following sections:

- Model—Use the Model tab to build and publish models:
 - “Understanding Cubes, Dimensions, and Measures Data Types” on page 78
 - “Working with Measure and Constraint Formulas” on page 82
 - “Data Flow and Mapping” on page 85
 - “Understanding the Model Tab User Interface” on page 86
 - “Building a Model” on page 87
 - “Reviewing and Publishing Model Objects” on page 108
 - “Changing Models After Publishing” on page 109

See “Using the Model Tab” on page 78.

- Presentation—Use the Presentation tab to manage models by:
 - Managing analysis types
 - Managing queries
 - Managing report templates
 - Reviewing changes and publishing model objects
 - “Changing Models After Publishing” on page 120
 - Managing worksheets
 - Managing workbooks

See “Using the Presentation Tab” on page 111.

- Administration—Administration tasks include managing the following areas:

- [Essbase connections](#)
- [Job queue](#)
- [Script editor](#)
- [Script templates](#)
- [Security filters](#)
- [System and application resources](#)
- [Users and Groups](#)

See [“Using the Administration Tab”](#) on page 127.

Using the Model Tab

Subtopics

- [Understanding Cubes, Dimensions, and Measures Data Types](#)
- [Working with Measure and Constraint Formulas](#)
- [Data Flow and Mapping](#)
- [Understanding the Model Tab User Interface](#)
- [Building a Model](#)
- [Reviewing and Publishing Model Objects](#)
- [Changing Models After Publishing](#)

Understanding Cubes, Dimensions, and Measures Data Types

Cubes

A cube is a subset of data organized in a multidimensional structure. Cubes are defined by a set of dimensions. A cube has at least two dimensions, one of which is a measure dimension. Other dimensions provide indexes to identify measure values in the cube.

A planning worksheet provides a view of the data in a cube. Each planning worksheet is associated with one cube. However, a cube can be associated with more than one planning worksheet, with each worksheet providing a view of a different slice of cube data.

The dimensions in a cube are based on categories of business data. Common dimensions: Fiscal, Manufacturing, Geography, Product, and Customer. Each dimension consists of members that can be organized into hierarchies with levels. For example, hierarchy levels in the Fiscal dimension include Months, Quarters, and Years.

Note: For additional examples of dimensions and members and an overview of how dimensions are used in planning worksheets, see [“Opening a Scenario in Excel”](#) on page 52.

Dimensions

- **Time**—A dimension composed of time periods such as Weeks, Months, Quarters, and Years. Time dimensions are organized into hierarchies that represent the fiscal or manufacturing calendar. You cannot edit time dimensions. Time dimensions are a special type of dense dimension.
- **Sparse**—A dimension in which a cube contains relatively fewer data values so that a smaller percentage of dimension intersections correspond to data values.
- **Dense**—A dimension in which a cube contains relatively more data values so that a greater percentage of dimension intersections correspond to data values.
- **Measure**—A dimension whose members are the measures calculated in an analysis. The measure dimension is not organized into hierarchies.

Dimension types:

A dimension can belong to more than one cube. Most cubes contain a time dimension such as Fiscal or Manufacturing.

A dimension definition contains the following elements:

- Members
- Namespaces
- Hierarchies

If hierarchies are defined, one hierarchy must be designated the default hierarchy. Hierarchies can be statically defined or determined dynamically at runtime. For example, hierarchies in the Components dimension depend dynamically on a Bill of Materials (BOM). As a result, multiple hierarchy roots might change during runtime. You cannot edit dimensions with dynamically defined hierarchies.

For editable dimensions, you can edit dimension hierarchies, namespaces, and members. After editing a dimension, you must restructure cubes that contain it to reorganize dimension members and recalculate data values. For example, adding a product to the Product dimension can affect other dimensions, such as Product Line, which must then be reorganized. Data values in the cube, at intersections of cube dimensions, also must be recalculated. Editing the measure dimension, for example, by modifying a measure formula, likewise requires recalculating data values.

Dimension Hierarchies

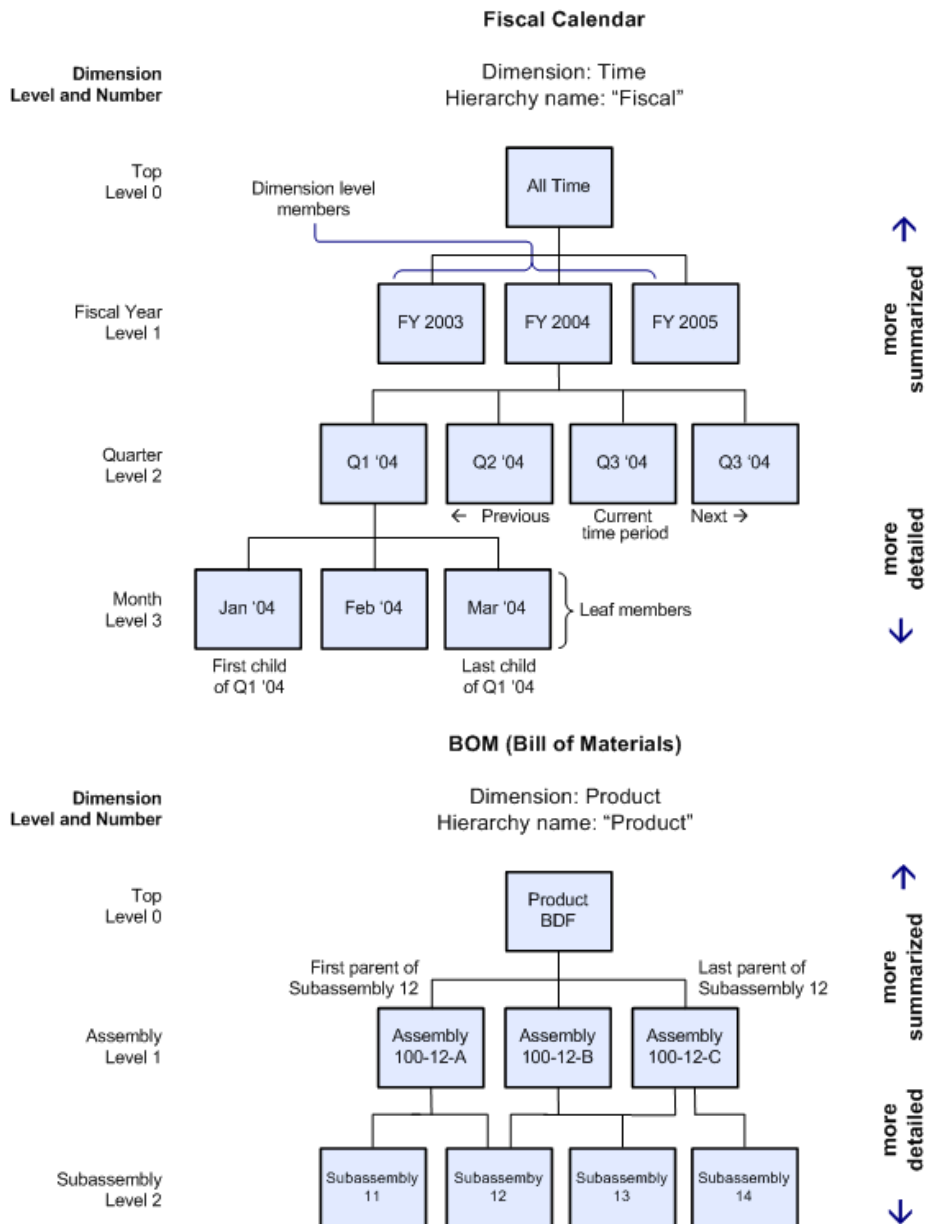
A dimension can contain one or more hierarchies, with one hierarchy defined as the default hierarchy. A hierarchy need not contain all dimension members, and a dimension member can appear in multiple hierarchies. In a hierarchy, each dimension member belongs to a hierarchy level.

Members that occupy the same level are siblings. The sequence of siblings in a level is important, making it possible to specify a particular sibling member in formulas for measures and exceptions. In a hierarchy diagram, a sibling sequence moves from left to right in a level, with the leftmost member the “first” member and the rightmost member the “last” member. In the

Administration section of the user interface, the sequence of siblings moves from top to bottom in a list, with the member at the top the “first” and the member at the bottom the “last.”

See [“Working with Measure and Constraint Formulas”](#) on page 82.

The following diagrams illustrate the terminology for dimension hierarchies.



Dimension Namespaces

A namespace is a mapping that matches dimension member names to internal system IDs. Each dimension member belongs to a dimension namespace. This mapping is especially important when dimension members share a name. For example, the Time dimension contains several members named “January”; however, each of these belongs to a different namespace corresponding to a fiscal year, such as FY 2009, FY 2004, or FY 2005. A dimension can contain multiple namespaces.

Likewise, in the Geography dimension, multiple dimension members may be named West, corresponding to the West territory in different geographic regions. Each dimension member West, therefore, belongs to a different namespace, such as NA or EMEA.

Note: A namespace is not the same as a dimension hierarchy. A hierarchy defines the structure of a dimension; that is, how the members are organized. A namespace provides a naming system to uniquely identify members. A member can belong to more than one hierarchy; however, a member belongs to only one namespace.

A dimension member has a fully qualified name, which includes its namespace. The fully qualified member name is expressed as follows:

`dimension name/namespace/member name`

For example, the fully qualified member name of the dimension member January in the dimension Fiscal in the namespace FY 2005 is expressed as:

`Fiscal/FY 2005/January`

A dimension has a default namespace whose name is the same as the dimension name. In addition, the default namespace contains a member with the same name as the dimension. For example, in the Fiscal time dimension, the fully qualified name of the default dimension member:

`Fiscal/Fiscal/Fiscal`

Measures

A measure is a member of a special type of dimension called a measure dimension. A measure dimension and its measures belong to only one cube. An example of a measure dimension is the Demand dimension, which belongs to the Demand cube and contains measures such as Projected Sales and Booked Units. A measure dimension is not organized into hierarchies and contains only one namespace with the same name as the cube to which it belongs.

A measure, for example, Projected Sales, has numerical values located at intersections of cube dimensions, with coordinates given by one member from each dimension. All measures have a data type of *double*.

Measure Types

Measure types:

- **Loaded**—Measure values are numerical data loaded from external systems or files into the Integrated Operational Planning database; for example, CSV files. Examples of loaded measures are Booked Sales and Shipped Units. When a loaded value does not exist, a formula can be used to derive it. The formula never overrides a loaded value.

For example, consider a formula in which the Inventory value for the current week is given by the value for the previous week. If loaded values for Inventory include a value for the current week, the loaded value is used. If a loaded value does not exist for the current week, the value for the previous week is used.

- **Input**—Measure values are numerical data entered directly into an Integrated Operational Planning planning worksheet. For example, one source of input measure values can be data in existing Excel spreadsheets used for forecasting.
- **Derived**—Measure values are calculated from formulas using Integrated Operational Planning built-in functions. Derived values can be based on other measures, including loaded and input measures. Examples of derived measures: Actual Average Selling Price and Excess Material Cost.
- **Cross-cube**—Measure values in one cube are calculated from formulas that depend on measure values in other cubes. Examples of cross-cube measures: Forecast Projected Sales and Forecast Booked Sales, whose values depend on measures in the Demand and Forecast cubes.

Key Metrics

A key metric corresponds to a measure value of interest; for example, a value of Projected Sales, that is located at a specified intersection of cube dimensions.

For example, the measure Projected Sales belongs to the Demand cube, with dimensions Demand, Product, and Fiscal. A key metric can be defined as the value of Projected Sales (in the Demand dimension) for product KFB (in the Product dimension) during the time period Q2 '05 (in the Fiscal dimension). A different key metric can be defined as the value of Projected Sales for All Products during the time period Fiscal 2005.

A scenario includes default key metrics whose values are calculated and displayed in the Key Metrics panel on the Perform Analysis screen or the Impact window in Excel during an analysis. Key metrics are chosen based on the measure values that are most important in an analysis type.

Working with Measure and Constraint Formulas

Subtopics

- [About Measure Formulas](#)
- [Measure Summarization](#)

In Excel, you can change display options to show the formulas used to calculate measures in a planning worksheet.

As a planner, you can view formulas to see how the measures you work with are derived.

As an administrator, you can customize planning worksheets to add measures, along with formulas for calculating their values from other data in the Integrated Operational Planning database.

The arguments (inputs) to functions used in formulas may be numbers, alphanumeric characters, variables, logical values (true or false), cell locations, or other formulas.

See:

- [Appendix B, “Writing Formulas”](#)

- [Appendix C, “Functions”](#) for detailed information on the Integrated Operational Planning calculation functions
- [Appendix D, “MDX Extensions”](#)

About Measure Formulas

Planning worksheets provide a view of measure values stored in cubes in the Integrated Operational Planning database. Measure values can be loaded, input, or calculated, depending on their original data source.

Calculated measure values are derived from formulas that take into account some or all of the following:

- **Cell Location**—The location in the worksheet of the cell containing the measure value.
- **Hierarchy Level**—The level to which a dimension hierarchy has been expanded or collapsed, which determines whether the measure represents a detailed or summarized value.
- **Time Period**—The time period over which a measure value applies. Calculated and uncalculated measure values can also be summarized using aggregate functions.

Measure Summarization

Measure values in a dimension can be summarized using aggregate functions to obtain summary values for different levels in a hierarchy. For example, in the Fiscal dimension, measure values that apply over a month can be summarized at the quarter level and again at the fiscal year level. In this way, a dimension hierarchy in a planning worksheet can be expanded or collapsed to expose more detailed or summarized data.

Measure values can be summarized differently in each dimension. Measure values cannot be summarized in the measure dimension.

The following aggregate functions can be used to summarize measure values in a dimension, depending on the type of dimension.

Count

Count returns a count of the values in the next lowest level of a dimension hierarchy.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy: April, May, and June. The summary value for Q2 '05 is therefore three. This function is used in dense and time dimensions only.

max

Max returns the highest value in the next level down in a dimension hierarchy; that is, the maximum value of the children in the hierarchy.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy: April, May, and June. If values for April, May, and June are 14, 11, and 15 respectively,

the summary value for Q2 '05 is 15. This aggregate function is used only in dense and time dimensions.

Min

Min returns the lowest value in the next level down in a dimension hierarchy; that is, the minimum value of the children in the hierarchy.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy: April, May, and June. If values for April, May, and June are 14, 11, and 15 respectively, the summary value for Q2 '05 is 11. This aggregate function is used only in dense and time dimensions.

Sum

Sum returns the sum of the values in the next level down in a dimension hierarchy; that is, the sum of the values of the children in the hierarchy.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy: April, May, and June. If values for April, May, and June are 4, 1, and 5 respectively, the summary value for Q2 '05 is 10. This aggregate function summarizes values in sparse, dense, and time dimensions.

Avg

Avg returns the average of the values in the next level down in a dimension hierarchy; that is, the average value of the children in the hierarchy.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy: April, May, and June. If values for April, May, and June are 4, 3, and 5 respectively, the summary value for Q2 '05 is 4. This function is used in dense and time dimensions only.

Last_in_Period

Last_in_Period—Used only for time dimensions (Fiscal and Manufacturing). Returns the value for the last member in a sequence of children.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy arranged in the sequence April, May, June, with values 4, 3, and 5 respectively. The summary value for Q2 '05 is the value for June (5). This function is used in dense and time dimensions only.

First_in_Period

First_in_Period—Used only for time dimensions (Fiscal and Manufacturing). Returns the value for the first member in a sequence of children.

For example, in the Fiscal dimension, the dimension member Q2 '05 has three children in the hierarchy arranged in the sequence April, May, June, with values 4, 3, and 5 respectively. The

summary value for Q2 '05 is the value for April (4). This function is used in dense and time dimensions only.

In dense and time dimensions, measures can be summarized using any of the previously described aggregate functions. In sparse dimensions, measures can be summarized only using the sum aggregate function.

See [“Understanding Cubes, Dimensions, and Measures Data Types”](#) on page 78.

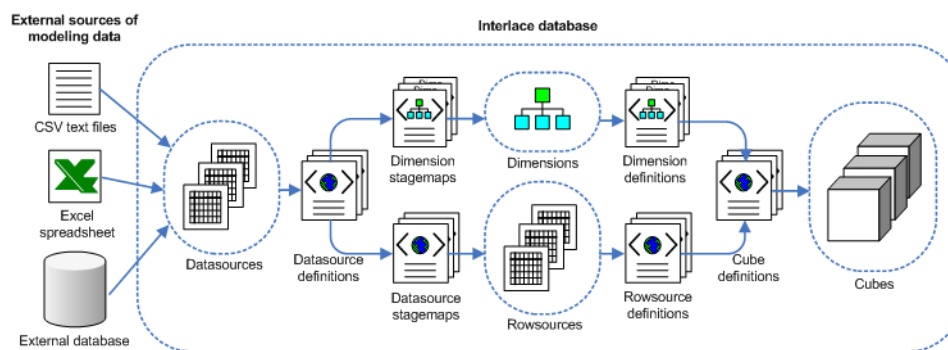
Data Flow and Mapping

You can model data in these ways:

- Create a model using the Model tab in the Administration Workbench
- Import a model using XML metadata definition files

Figure 10 illustrates the relationship and flow between various modeling elements.

Figure 10 How Modeling Data is Staged and Loaded into Integrated Operational Planning



- **Data Sources**—Relational tables in the Integrated Operational Planning database where copies of external modeling data are stored. By default, the Integrated Operational Planning application looks in `not/install/config/data` for files containing the data; for example, CSV text files. The data is then loaded into data sources in the Integrated Operational Planning database. The default directory for external modeling data is specified in `ISServer.properties`.
- **Data Source Definitions**—XML files that specify which data source columns in a data source to use. Using all columns in a data source is not required. By default, data source definition files are in `/install/custom/data source`.
- **Stagemaps**—XML files that map data sources to row sources or dimensions in the Integrated Operational Planning database. One data source can be mapped to multiple row sources. For example, each row source can contain some columns of the original data source. By default, stagemap files are in `directory/install/custom/loader`.

When data sources are mapped to dimensions, Integrated Operational Planning maps the names from the tabular data source to the elements of the dimension's hierarchy. In addition to mapping Data source columns to dimension members at all levels, Integrated Operational

Planning maps the attributes of dimension members and specifies the relationship (for example, parent-child).

- **Row Sources**—Relational tables in the Integrated Operational Planning database containing data values. Row sources can be mapped to cubes.

Integrated Operational Planning supports one-to-one mapping of row source values to one value in a cube measure or aggregation of row source values to one value in cube measure. Row source columns are mapped to dimension member names in the cube, and the values are mapped to different measures.

- **Dimension Definitions**—XML files that specify the namespace name, hierarchy names, and hierarchy level names for a dimension. By default, dimension definition files are in `/install/custom/model`.
- **Row Source Definitions**—XML files that specify the columns in a row source and the parent-child relationships between row sources. By default, row source definition files are in `/install/custom/row source`.
- **Cube Definitions**—XML files that specify the dimensions in a cube, the names of measures in the cube and measure formulas, the names of row sources mapped to the cube, and cube-to-cube mapping information. By default, cube definition files are in `/install/custom/model`.

The mapping among cubes allows data values in one cube to flow to data values in another cube. The cubes have different structures, and the mapping defines the data flow. The mapping is like a table join in the relational world—it defines the common dimensions and how to map or join them.

Understanding the Model Tab User Interface

Sections in the Model tab:

- **Object Browser**—Shows the selected elements. You can view data sources, row sources, dimensions, and cubes. Click **Actions** to add, delete, or edit an element.
- **Object Properties**—Properties of the selected element
- **Object Details**—Details of the selected element. Select a tab to define how to view the details.
 - **Definition**—Name, description, type, file type, file name, and data fields for the selected element.
 - **Graph**—Graphical representation of the relationship between the selected element and other elements

Click an element to view available actions for the element. For example, clicking a cube opens a menu, where you can center, edit, or delete the cube; map the cube to another data source; or map the cube to another cube.

Click the line between elements to edit or delete the mapping between the elements. Available mappings: cube-to-cube, cube-to-row source, and row source-to-data source.

- **Relationships**—Name, type, and mapping for the selected element

Building a Model

Subtopics

- [Creating a Data Source](#)
- [Creating a Row Source](#)
- [Creating a Dimension](#)
- [Creating a Cube](#)
- [Creating Maps](#)

Use the Model tab to build a model.

Creating a Data Source

► To create a data source:

- 1 On the Object Browser **View** menu, select **Data Sources**.
- 2 Click **Actions** and select **Add**.
- 3 In the Data Source Wizard, enter data source **Properties** information.
 - **Name**—Data source name
 - **Description**—Description of the data source
 - **Type**—Flat File, Excel File, Database, or Essbase
 - **Create Row Source and Stagemap**—Select to automatically create a row source and a stagemap from the data source. The default value is set by the `datadesigner.default.rowsource.create` parameter in `ISServer.properties`.
 - **Details**—Information about the file
 - In **Flat File**, enter a file name and a file type (delimited or fixed width).
 - In **Excel File**, enter a file name and an Excel sheet name.
 - In **Database**, define a database connection.
 - In **Essbase**, select an Essbase connection or click **New** to define a connection.

When entering a file name for Flat Files or Excel Files, perform an action:

 - Enter the file name.
 - Select **Browse** to display the files on the server data tray.
 - **Upload** to upload a file from your computer. When you select a file after clicking **Upload**, the file is uploaded to the server.
- 4 Enter data source **Configuration** information and click **Preview**.

In **Flat File**, enter:

- **Delimiter**—Separator between fields in a data source. Select Tab, Comma, Semicolon, or Pipe.

- **Start Row**—Row on which to start importing data (The Start Row is usually 2.)
- **Header Row**—Row that contains the column titles (The Header Row is usually 1.)
- **Missing Trailing Values**—How to treat rows or columns that are missing values. Select **Treat as Error** or **Treat as Null**.
- **Text Qualifier**—Whether to display quotation marks around text values:None, Double Quote, or Single Quote.
- **End Row**—Row on which to stop importing the data (Leave blank to import all rows.)
- **Row Increment**—Number of rows to advance when importing data (Default is 1)
- **Pivot Data**—If the underlying Excel data is tabular, select False. If the underlying Excel data is in a pivot format, select True. Pivoting implies that the file is anchored vertically along a few columns and then horizontally along other columns, with data at the intersection of the vertical and horizontal columns.

For **Excel File**, enter:

- **Start Row**—Row on which to start importing data (Usually 2.)
- **Header Row**—Row that contains the column titles (Usually 1.)
- **Missing Trailing Values**—How to treat rows or columns that are missing values. Select **Treat as Error** or **Treat as Null**.
- **End Row**—Row on which to stop importing the data (Leave blank to import all rows.)
- **Row Increment**—Number of rows to advance when importing data (Default is 1)
- **Pivot Data**—If the underlying Excel data is tabular, select False. If the underlying Excel data is in a pivot format, select True. Pivoting implies that the file is anchored vertically along a few columns and then horizontally along other columns, with data at the intersection of the vertical and horizontal columns.
- **Skip Hidden Rows**—Whether to skip hidden rows.

For **Database**, enter the SQL for the data source configuration.

For **Essbase**, enter:

- **Query Type**—Report Script or MDX
- **Query**—Query to be sent to Essbase

Integrated Operational Planning internally flattens the results returned from Essbase and displays the results under **Data Source Preview**.

5 Review the data **Fields**.

- **Refresh Fields**—Refresh the defined data fields. Click this button if no fields are displayed.
- **Create**—Create a data field

Enter this information:

- **Name**—Data field name

- **Type**—String, Double, Integer, Date, or Duration
- **Nullable**—Whether a value can be empty. True or False.
- **Orientation**—For pivot formatted data, select Vertical or Horizontal to define how to display the data field in relation to other data fields

If vertical, enter the column in which the data field will be displayed.

If horizontal, enter a column range, row number and row type. A row type can be absolute or relative. Absolute rows are horizontal header columns, and you must specify the actual row numbers. Relative rows indicate the data columns, and the row number must be zero.

- **Default**—Default data field value
- **Date Format**—For date fields, select the format in which the date will appear.
- **Null Values**—How to treat null values. Select **Remain as Null** or **Default to Previous Non-Null Value**.
- **Edit**—Edit a data field
- **Delete**—Delete a data field

6 Click **Save** to create the data source and map it to a row source.

The data source is displayed in the Object Browser and is automatically mapped to a row source.

See [“Data Flow and Mapping” on page 85](#) for details on how data is staged and loaded into the Integrated Operational Planning database.

Tip: Review the row source generated through automatic mapping for key values. The automatic generation assigns the first column as the key column of the row source. If this assignment is not valid, you must modify the key columns of the row source. Similarly, when the stagemap is automatically generated for the mapping between a data source and a row source, a default group name is assigned to the stagemap. Review this default group name for accuracy.

Creating a Row Source

➤ To create a row source:

- 1 On the Object Browser **View** menu, select **Row Sources**.
- 2 Select **Actions**, and then **Add**.
- 3 In the Row Source Wizard, enter row source **Properties**.
 - **Name**
 - **Description**
 - **Type**—Application, Custom, or Join

- **Parent Row Source** (*application* row sources only)—Select a parent row source, or select None if there is no parent.
- **Time Varying**—For *application* row sources, select True or False to specify whether the data will change with time. If True, an `effectiveTime` column is created in the row source.

The `effectiveTime` column is not visible in the Row Source Wizard; however, it must be mapped in the stagemap where the row source is involved.

To map `effectiveTime` in the stagemap:

- In the Graph tab in the Model, click the line between the row source and the data source and select Edit Mapping.
- In Column Maps, select Advanced beside the `effectiveTime` column, and enter `$begin(Fiscal)`, where *Fiscal* is the name of a time dimension in the model.
- **Generate Change Entries** (*custom* row sources only)—If the row source is mapped to a cube, select True. If the row source is *not* mapped to a cube, select False.
- **Check Table Exists** (*custom* row sources only)—When a row source is imported, Integrated Operational Planning checks for the existence of an external table. If the external table does not initially exist, an error is returned and the import fails. To avoid this error, set the value to False. For strict error checking and for situations where the external table exists from the beginning, set the value to True.

4 Create row source **Columns**.

- To create *application* row source columns, click **Create** and perform an action:
 - **Add Columns from the Data Source**—Select **Add from Data Source** to display the Add Row Source Columns dialog box. When you select a data source, the columns in the data source are displayed. Select the columns and click OK.
 - **Create New Columns**—Click **Create New** to create a column. Enter the following information:
 - ❑ **Name**
 - ❑ **Type**—String, Double, Integer, Date, Timestamp, or Boolean
 - ❑ **Nullable**—Whether a value can be empty. Select True or False.
 - ❑ **Data Constraint**—Whether the data is restricted. Select None, Range, or Specific Values. If you select Range, enter the starting and ending values. If you select Specific Values, enter a list of values. To add a value, click Add.
 - ❑ **Style**
 - ❑ **Size**—(String) Column size
 - ❑ **Precision**—(Double) Accuracy in terms of number of decimal places. For example, with a precision of 3, the number 45.3768 is rounded to 45.377.
 - ❑ **Date Format**
 - ❑ **Default**—Default column value

- To create *custom* row source columns, enter a database table name, click Refresh Fields and edit the following information:
 - **Name**
 - **Nullable**—Whether a value can be empty. Select True or False.
 - **Default**—Default column value
 - **Style**
- To create *join* row source columns, enter the following row source configuration information and click Refresh Fields:
 - **Primary Row Source**
 - **Secondary Row Source**
 - **Join Type**—Inner or Left Outer

After you click Refresh Fields, row source columns based on the row source configuration information are generated. To define columns, select corresponding Select Column check boxes or select **include all primary row source columns in select columns** or **include all secondary row source columns in select columns**.

After you define select columns, you can define join columns from the secondary row source.

5 Enter Key and Indices information.

- **Row Source Key**—Each row has a unique key. Click **Edit** and define the row source key by moving the desired column(s) from Available Columns to Selected Columns. (For *time varying* row sources, the effective date is automatically part of the key.)
- **Row Source Mapping Order**—Column order in the row source. Click **Edit** move the desired columns from Available Columns to Selected Columns.
- **Row Source Indices**—For *application* and *custom* row sources, click **Create** or **Edit** remove the desired columns from Available Columns to Selected Columns to define row source indices. Select **True** or **False** to define whether the row source index is unique. You can define as many row source indices as desired. Defining row source indices improves query performance.
- **Row Source Index Details**—For *application* and *custom* row sources, click a row source index to view its details.

6 Click **Save** to create the row source.

The row source is displayed in the Object Browser.

Note: See “Data Flow and Mapping” on page 85 for details on how data is staged and loaded into the Integrated Operational Planning database.

Creating a Dimension

► To create a dimension:

1 On the Object Browser **View** menu, select **Dimensions**.

2 Click **Actions** and select **Add**.

3 In the Dimension Wizard, enter dimension **Properties**.

- Name
- Description
- Type—Dense, Sparse, or Time

See [Dimensions](#).

4 Enter dimension **Hierarchies** information.

Enter a dimension hierarchy **Name**, select a **Hierarchy Type**, and enter hierarchy details.

- For **Static** hierarchies, enter the following information:
 - **Root Member**—Name and Display Name for the top member of the hierarchy
 - **Levels**—Number of levels in the hierarchySelect **Associate Styles to Levels** to define styles for the levels in the hierarchy
- For **Dynamic** hierarchies, enter the following information:
 - **Class**—Java class that defines the hierarchy
- For **Row Source Based** hierarchies, enter the following information:
 - **Row Source Name**
 - **Parent Name**
 - **Parent Display Name**
 - **Parent Description**
 - **Parent Attributes**
 - **Child Name**
 - **Child Display Name**
 - **Child Description**
 - **Child Attributes**

In a row source-based hierarchy, parent and child members can have many relationships. An example of a row source based hierarchy is a Bill of Materials.

5 Enter dimension **Attributes** .

Click **Create** or **Edit**, and enter attribute details. Attributes define information about a member in the dimension. Attributes are used for filtering and searching.

The maximum number of dimension attributes a model can have is 20. This information is stored in the `IS_DIMENSION_ATTRIBUTES` table in the Integrated Operational Planning database.

6 Click **Save to create the dimension.**

The dimension is saved in the Object Browser.

Note: See [“Understanding Cubes, Dimensions, and Measures Data Types” on page 78](#) and [“Data Flow and Mapping” on page 85](#) for additional information on dimensions and how data is staged and loaded into the Integrated Operational Planning database.

Creating a Cube

Subtopics

- [Creating Measures](#)
- [Editing Measures](#)
- [Creating Constraints](#)

► To create a cube:

1 On the Object Browser **View menu, select **Cubes**.**

2 Click **Actions, and select **Add**.**

3 In the Cube Wizard, enter **Properties information.**

- Name
- Description
- Cube Publish Order

Integrated Operational Planning automatically assigns a cube order. To accept this order, leave the Cube Publish Order at *default*; otherwise, assign a Cube Publish order.

4 Define cube **Dimensions.**

Click **Add** and select dimensions from the Add Dimensions dialog box. After you add a dimension, you can select it to view its details.

Note: See [Dimensions](#) and [“Creating a Dimension” on page 92](#).

5 Define cube **Measures.**

Click **Create** and enter measure information.

Note: See [“Measures” on page 81](#) and [“Creating Measures” on page 94](#).

6 Review dimension **Summarization information.**

Click a dimension to view details.

7 Define cube **Constraints.**

Click **Create** to access the Constraint Wizard and define a constraint. After you add a constraint, you can select it to view its details.

See [“Creating Constraints” on page 97](#).

8 Define cube **Allocation Maps**.

Click **Create** to access the Allocation Map Wizard and define an allocation map.

See [“Creating and Editing Allocation Maps” on page 107](#).

9 Click **Save** to create the cube.

The cube is displayed in the Object Browser.

Note: See [“Understanding Cubes, Dimensions, and Measures Data Types” on page 78](#) and [“Data Flow and Mapping” on page 85](#) for additional information on dimensions and how data is staged and loaded into the Integrated Operational Planning database.

Creating Measures

► To create a measure:

1 On the **Measures** page in the Cube wizard, click **Create** and enter the following information:

- **Name**
- **Display Name**—How the measure is displayed in Excel
- **Type**—Derived, Input, Loaded, or Cross-cube
See [Measure Types](#).
- **Precision**—Accuracy in terms of number of decimal places. For example, with a precision of 3, the number 45.3768 is rounded to 45.377.
- **Check Consistency**—Whether Integrated Operational Planning should check whether the measure value is different from the calculated value. Not valid for input values from another source.
- **Hidden**—Whether the measure is hidden from users
- **Style**—Formatting

Click **Create** again to enter additional measures.

2 Click a measure number to enter details.

You can enter measure details, or you can copy measure details from another measure and edit. To copy measure details from another measure, select a measure next to **Copy From**. You can copy measure details only from another measure *of the same type*. For example, if the current measure is a derived measure, Copy From displays only derived measures.

- For *Derived* measures, enter the following information:
 - **Description**
 - **Measure Formula**—Enter a measure formula and click **Validate**.

Note: See [“Working with Measure and Constraint Formulas” on page 82](#).

- **Summarizations**—Dimension summarization information. For each dimension, you can select a rollup (complex aggregation operator) or a roll-down (allocation operator) value.

If you select a rollup or roll-down value, click **Add or Edit Summarization Details** and enter the **Source Level**. For rollup, this level is the level from which the aggregation starts. For roll-down, this is the level from which allocation starts.

For *custom* values, enter the following additional information:

- **Operation Map**—An allocation map defined in the cube where you define intermediate end levels
- **Formula**—Batch calculation formula used during rollup or roll-down
- **On Change Measure**—Dependency measure. Changing this measure triggers rollup or roll-down.
- **On Change**—Interactive equivalent of batch formulas for rollup or roll-down
- **Propagate on Existence**—Whether to create new children and roll down values to them

See:



- [“Measures” on page 81](#)
- [“About Measure Formulas” on page 83](#)
- [“Measure Summarization” on page 83](#)
- For *Input* measures, enter the following information:

- **Description**
- **Summarizations**—For each dimension, you can select a roll-up (complex aggregation operator) or a roll-down (allocation operator) value.

See the *Summarizations* bullet under the *For Derived Measures* bullet for information on what to enter if you select a rollup or a roll-down value.

With input measures, you can input the measure values at runtime while performing analysis.

- For *Loaded* measures, enter the following information:

- **Description**
- **Row Source**—Click  and select a row source
- **Row Source Column**—Click  and select a row source column
- **Measure Formula**—Enter a measure formula and click **Validate**.

See [“Working with Measure and Constraint Formulas” on page 82](#).

- **Summarizations**—Dimension summarization information. For each dimension, you can select a rollup (complex aggregation operator) or a roll-down (allocation operator) value.

See the *Summarizations* bullet under the *For Derived Measures* bullet for information on what to enter if you select a rollup or a roll-down value.

- **Loaded Measure Filters**—How to filter the measures
 - Click **Add or Edit Loaded Measure Filters**.
 - Click **Create or Edit**.
 - Enter filter details by selecting a column, selecting an operator, and entering a value. (For example, the column could be *customerid*, the operator could be *equals*, and the value could be *100*.)

- For *Cross-Cube* measures, enter the following information:

- **Description**
- **Cube**—Source cube containing the cross-cube value
- **Measure**—Measure in the source cube on which the current target measure depends
- **Measure Formula**—Formula used for batch calculation. Enter a measure formula and click **Validate**.

See [“Working with Measure and Constraint Formulas” on page 82](#).

- **OnChange Formula**—Formula used for interactive calculation
- **Summarizations**—Dimension summarization information. For each dimension, you can select a roll-up (complex aggregation operator) or a roll-down (allocation operator) values

See the *Summarizations* bullet under the *For Derived Measures* bullet for information on what to enter if you select a rollup or a roll-down value.

3 Enter a default measure.

The first measure created is automatically assigned as the default measure. Click **Change Default Measure** and select a different measure.

See:

- [Appendix B, “Writing Formulas”](#)
- [Appendix C, “Functions”](#)
- [Appendix D, “MDX Extensions”](#)

Editing Measures

► To edit a measure:

- 1 On the **Measures** page in the Cube wizard, click **Edit** and change the measure information. For measure information, see

- [“Creating Measures” on page 94](#)
- [Appendix B, “Writing Formulas”](#)
- [Appendix C, “Functions”](#)
- [Appendix D, “MDX Extensions”](#)

2 Click **OK**.

Creating Constraints

► To create a constraint:

1 In the **Administration Workbench**, go to the **Model** tab and select **Cubes**.

2 In the **Object Browser**, select **Cubes** and do one of the following:

- To create a cube and constraints, click **Actions**, and then **Add**.
- To edit a cube to create constraints, select a cube, then click on **Actions**, then **Edit**.

3 In the **Cube wizard**, click **Constraints**, and then click **Create**.

A **Constraint Wizard** is displayed.

4 Enter **Properties** information.

- **Name**—Constraint name
- **Display Name**—How the constraint is displayed in Excel
- **Description**
- **Type**
 - **Reporting (Batch only)**—Business-rule violations are evaluated only during batch calculation and not during interactive what-if analysis.
 - **Display Only**—Business rule violations are not evaluated up front; instead, Integrated Operational Planning checks for these violations at certain intersections and flags them in the user interface. Violations are not stored.
 - **Regular (Batch and Interactive)**—Business-rule violations are evaluated for batch calculations and interactive calculations.

For example, violations may be introduced during what-if analysis, or existing violations may get fixed by changing some values.
- **Default Assignee**—Whom to notify to fix the violation
- **Priority**—Priority of the constraint. Select Low, Normal, or High
- **Days till Due**—Number of days until the default assignee should fix the violation

5 Define the constraint **Condition**.

To define the constraint condition, select a measure and enter a formula. Click **Validate** to ensure that the formula is valid before proceeding.

See [Appendix B, “Writing Formulas”](#), [Appendix C, “Functions”](#)

[“About Measure Formulas” on page 83,](#)

For example, a component shortage constraint condition for an ending inventory:

```
Component_Shortage."Ending Inventory"[level(Month)] assert once (isPast() or "Ending Inventory" >= 0)
```

6 Define constraint **Reporting Detail**.

Click **Create** and enter the following information:

- **Name**
- **Display Name**—How the constraint is displayed in Excel
- **Type**—String, Double, Integer, Date, Timestamp, or Boolean
- **Size**—(String) Constraint size
- **Precision**—(Double) Precision of the constraint
- **Format**—(Date) Format in which the constraint date is displayed
- **Key**—Whether the constraint has a key associated with it. Select True or False.
- **Formula**

7 Define the **Severity** of the constraint.

The severity is a formula that gets evaluated and is used with the styles.

For example, the following formula evaluates “severity” to a number. The number is then used to flag the violation in different colors (specified in styles).

```
<Severity><![CDATA[
  (75 - "ASP Attainment Pct")/75*100
]]></Severity>
<Styles>
<Style styleID="greenBackground" layer= "1"><![CDATA[
Severity <= 50
]]></Style>
<Style styleID="orangeBackground" layer= "1"><![CDATA[
Severity > 50 and Severity < 75
]]></Style>
<Style styleID="redBackground" layer= "1"><![CDATA[
Severity >= 75
]]></Style>
</Styles>
```

8 Define the **Style** of the constraint.

Click **Create** and enter the following information:

- **Style**—Style to apply to the constraint when it is displayed
- **Layer**—Violation precedence

For example, violations associated with styles with a higher layer number take precedence over violations associated with styles with a lower layer number.

- **Formula**—Formula to evaluate whether to use a particular style. The severity is often used in the formula.

9 Click **Save**.

Creating Maps

Subtopics

- [Creating Cube-to-Cube Maps](#)
- [Creating a Row Source-to-Cube Map](#)
- [Creating Data Source-to-Row Source Maps](#)
- [Creating Data Source-to-Dimension Maps](#)
- [Creating and Editing Allocation Maps](#)

When you add a data source, row source, dimension, or cube in the Model, maps are created. Available mappings include cube-to-cube, row source-to-cube, data source-to-row source, data source-to-dimension, and allocation maps.

- To edit the mapping, click the line between objects in the Object Details Graph tab and select **Edit Mapping**.

When you edit a map, a green check mark appears beside the object in the Object Browser.

Creating Cube-to-Cube Maps

- To create a cube-to-cube map:

- 1 In the **Model**, go to the **Object Details Graph** tab, click the line between the cubes, and select **Edit Mapping**.

A Cube Mapping Wizard is displayed, where you can edit information about the mapping.

- 2 Review **Source and Target** cube information.
- 3 Define the **Cube Scope** for the source and target cubes by selecting a scope type and entering a value for each dimension.

The cube scope defines which members participate in the data flow from the source cube to the target cube.

- **All**
 - **Function**—Define a custom function; which, when evaluated, returns a list of members that participate.
 - **Level**—Select a level in the dimension hierarchy. The members in that level are the members that participate.
 - **Member**—Explicitly specify the members that participate.
- 4 Review or edit **Dimension Map** information for the source and target cubes.

To view detailed information for a dimension, click the dimension name. The information is displayed under Dimension Map Details. To change dimension map information, click **Edit**.

- **Dimension**
- **Namespace**—Name to uniquely identify dimension members

When creating a dimension map, Integrated Operational Planning passes the name of the dimension and the value of the namespace to the mapping function to evaluate the target members.

In the following example, when Integrated Operational Planning evaluates the mapping function, it uses the "ProductLine" argument as well as the namespace and dimension values to determine the target dimension members.

```
dimension="Product"  
namespace="ESG"  
mapping="oneToOneMap(ProductLine) "
```

The namespace can be the same as the dimension name, but it need not be. See [Dimension Namespaces](#) for detailed information on namespaces.

- **Transformation**—An optional function applied to the result of the mapping function
Integrated Operational Planning first maps source members to target members using the mapping function (`oneToOneMap(ProductLine)` in the previous example). If the target members are unavailable after mapping is complete, the transformation information is used. Transformation information is applied in the context of the target dimension.

See [“Member Functions” on page 202](#).

The following is a sample dimension map:

```
dimension="Manufacturing"  
namespace="Manufacturing"  
mapping="calendarMap(Fiscal) "  
transformation="childMembers"
```

- **Mapping Type**—Simple, Advanced, or Fixed Member
 - For Simple mappings, define the type of mapping and the dimension to which to map. See:
 - [“Member Name Mapping Functions” on page 197](#):
 - [“exactMap” on page 197](#)
 - [“oneToOneMap” on page 197](#)
 - [“Time Dimension Mapping Functions” on page 194](#):
[“calendarMap” on page 194](#)
 - For Advanced mappings, click **Adv** and select the mapping types. See:
 - [“Member Name Mapping Functions” on page 197](#):
 - [“exactMap” on page 197](#)

- “oneToOneMap” on page 197
- “Time Dimension Mapping Functions” on page 194:
 - “calendarMap” on page 194
- “Dimension Member Attribute Name Mapping Functions” on page 195:
 - “attributeMap” on page 195
 - “attributeReverseMap” on page 196
- “Splicing and Splitting Mapping Functions” on page 199
- Appendix D, “MDX Extensions”:
 - “RowSourceLookup(*RowSourceColumn* [, *tuple*])” on page 236
 - “RSQlGenerateSet(*RSQlQuery* [, *dimHierarchy*, *NameColumn*, *NameSpaceColumn*] +)” on page 235
- For Fixed Member mappings, select the member.

5 Click **Save**.

Creating a Row Source-to-Cube Map

A Row Source-to-Cube map defines how row source columns are mapped to the fully qualified names of members in different cube dimensions. You can load data from each row source into cube cells.

In general, a single mapping exists between a row source and a cube; however, you can change the mapping by measure.

For example, assume that one measure, *shipped units*, maps the “shipped_date” column of the row source to the cube, but all the other measures use “order_date”. In this example, you would use normal mapping overall, and “custom” mapping for specific measures.

► To create a row source-to-cube map:

- 1 In the **Model**, go to the **Object Details Graph** tab, click the line between a row source and a cube, and select **Edit Mapping**.
- 2 In the Row Source Mapping Wizard, review **Source and Target** information.
 - **Row Source**—The source row source for which you are editing the mapping
 - **Cube**—The target cube for which you are editing the mapping
 - **Custom Mapping**—Whether to use custom mapping for each specific measure. Select True to use a Java class for the mapping; otherwise, select False.
 - **Partial Map**—Select True if all of the cube dimensions are not mapped to row source key columns; otherwise, select False.

Note: If the row source is time-varying, and all the dimensions except time are mapped, select False.

- **Summarized Value**—Select *True* to aggregate multiple values from a row source to single value in a cube. Select *False* for one-to-one mapping.

3 Define **Default Map** information.

For each cube dimension in the measure:

- Select a row source column.
- **Optional:** Select **Fixed Member** to add a member name.
- Select an attribute name.

If no attributes are assigned to the cube dimension, *none* is the only option for Attribute Name. You assign attributes to dimensions in [step 5](#) in the Dimension Wizard.

See “[Creating a Dimension](#)” on page 92.

- Enter a namespace, or select **Use Row Source Column as Namespace** and select a row source column from the menu.

If you defined custom mapping as *true* on the Source and Target wizard page, enter the following additional information:

- **Custom Mapper Class**—Class to generate the mapping from row source row to cube cell
- **Key Generator Class**—Class to generate the row source keys given a cube cell

4 Define **Measure Specific Maps**.

If you defined custom mapping as *false* on the Source and Target wizard page, to define measure-specific maps, click **Add Measure Map** and select a measure.

If you defined custom mapping as *true* on the Source and Target wizard page, you cannot define measure-specific maps.

5 Define **Filter** information.

Click **Create** and enter the following filter details:

- **Measure**—Select a measure or leave at All Measures to include all measures in the filter.
- **Column**—Select a column.
- **Value**—Enter a measure value.
- **Dimension**—Select a dimension.
- **Operator**—Select an operator.
- **Format**—Optional Format for the filter value. You can leave the Format blank.

To edit a filter, click **Edit**.

6 Click **Save**.

Creating Data Source-to-Row Source Maps

Note: A data source-to-row source map is also known as a *stagemap*. See stagemaps under “[Data Flow and Mapping](#)” on page 85.

➤ To create a data source-to-row source map:

1 From the **Model** tab, select the **Object Details Graph** tab, then click the line between a data source and a row source, and then select **Edit Mapping**.

2 In the Data Source Mapping Wizard, review **Source and Target** information.

- **Name**—Name used to identify the map (Assigned by Integrated Operational Planning)
- **Data Source**—Source data source for which you are editing the mapping
- **Row Source**—Target row source for which you are editing the mapping
- **Groups**—Used to combine the related set of row sources. Integrated Operational Planning defines all the stagemaps with a group name of *Bootstrap-RowSource-Stagemaps*. Using this model group, you can create without entering commands for each stagemap.

3 Define **Column Maps**.

For each target column, perform an action:

- Select a source column.
- Select **Advanced**, and then click **Add** to add source columns.

4 Define **Filters**.

Click **Create**, select a filter type, and then enter filter details:

- **Simple Filter**
 - **Data Source Field**—Field on which to filter the information
 - **Filter Operator**—Operator to use in the filter
 - **Filter Value**—Value for the filter. To include columns in the filter value, click **Add** and select a column.
 - **Non-matching are Errors**—Whether to log data values that do not match filter values in the error log file. Select **True** or **False**. (The default is **False**; data values that do not match filter values are not entered in the log file.)
- **Row Source Filter**
 - **Data Source Field**—Field on which to filter the information
 - **Row Source**—Row Source to use in the filter
 - **Row Source Column**—Row Source column to use in the filter
 - **Match**—Select **True** to allow only matching records in the filter. Select **False** to allow only nonmatching records in the filter.
 - **Non-matching are Errors**—Whether to log data values that do not match with filter values in the error log file. Select **True** or **False**. (The default is **False**; data values that do not match filter values are not entered in the log file.)
- **Dimension Filter**
 - **Data Source Field**—Field on which to filter the information
 - **Dimension**—Dimension to use in the filter

- Filter On—Select Namespace or Hierarchy. If you select Namespace, enter a value. (If you leave the Namespace value blank, the default hierarchy is used.) If you select Hierarchy, select a hierarchy and a hierarchy level.

See [Dimension Namespaces](#) and [Dimension Hierarchies](#).

- Allow Members—Select **True** to include only the records that exactly match the dimension member names. Select **False** to include only the records that do not match dimension member names.
- Non-matching are Errors—Whether to log data values that do not match filter values in the error log file. Select **True** or **False**. (The default is False; data values that do not match filter values are not entered in the log file.)
- **Distinct Filter**
 - Filter Value—Value for the filter. To include columns in the filter value, click **Add** and select a column.
 - Distinct—Whether data should have distinct values across the data set. Select **True** or **False**.
 - Non-matching are Errors—Whether to log data values that do not match filter values in the error log file. Select **True** or **False**. (The default is False; data values that do not match filter values are not entered in the log file.)
- **Custom Filter**
 - Class Name—Java class name that contains the custom filter logic
 - Parameters—Parameters passed to the Java class

5 Define Custom Processing information.

● Custom Pre-processing

To define custom pre-processing details, click **Create** and enter:

- **Class Name**—Java class containing preprocessing logic before staging starts for this stage map
- **Parameters**—Parameters passed to the preprocessing Java class

To define parameters, click **Add** and enter a value, type, and format for the parameter.

● Custom Post-processing

To define custom postprocessing details, click **Create** and enter:

- **Class Name**—Java class containing postprocessing logic after staging ends for this stage map
- **Parameters**—Parameters passed to postprocessing Java class

To define parameters, click **Add** and enter a value, type, and format for the parameter.

6 Click **Save**.

Creating Data Source-to-Dimension Maps

Note: A data source-to-dimension map is also known as a stagemap. See stagemaps under “[Data Flow and Mapping](#)” on page 85.

➤ To create data source-to-dimension maps:

1 From the **Model** tab, select the **Object Details Graph** tab, then click the line between a data source and a dimension, and then select **Edit Mapping**.

2 In the Data Source Mapping Wizard, review **Source and Target** information.

- **Name**—Name used to identify the map (Assigned by Integrated Operational Planning.)
- **Data Source**—The source data source for which you are editing the mapping.
- **Dimension**—The target dimension for which you are editing the mapping
- **Groups**—Used to combine the related set of row sources. Integrated Operational Planning defines all the stagemaps with a group name of Bootstrap-Row Source-Stagemaps. Using this model group, you can create without entering commands for each stagemap.

3 Define **Hierarchy Level Maps**.

The hierarchies defined as part of dimensions are displayed here. Use hierarchy level maps to:

- Define how to map the data source to members in different levels of the hierarchy.
- Map the dimension member attributes.
- Define the mapping for the attributes defined in the dimension.

To view details about a hierarchy level, click a hierarchy and review its details under **Hierarchy-Level Map Details**.

To edit a hierarchy level, select a hierarchy, click **Edit**, and define mapping information by either selecting a column or entering value for each of the following:

○ **Parent**—Parent level in the hierarchy

As an example of entering a value, assume that you have a parent level of `family_name`. Assume further that you want to combine two columns (`family_name` and `capacity`) and change the parent name. In this case, you would enter `${family_name}#${capacity}` as the value.

○ **Namespace**—Name to uniquely identify dimension members

See [Dimension Namespaces](#).

○ **Name Column**—Data source column that indicates the member name or an expression that combines data source columns

○ **Display Name Column**—Data source column that indicates the display name or an expression that combines data source columns

- **Description Column**—Data source column that indicates the description or an expression that combines data source columns
- **realName Attribute**—Attributes defined in the dimension

4 Define Filters.

Click **Create**, select a filter type, and enter filter details:

- **Simple Filter**

- **Hierarchy Level**—Hierarchy level on which to filter the information
- **Data Source Field**—Field on which to filter the information
- **Filter Operator**—Operator to use in the filter
- **Filter Value**—Value for the filter. To include columns in the filter value, click **Add** and select a column.
- **Non-matching are Errors**—Whether to log data values that do not match filter values in the error log file. Select **True** or **False**. (The default is **False**, data values that do not match filter values are not entered in the log file.)

- **Row Source Filter**

- **Hierarchy Level**—Hierarchy level on which to filter the information
- **Data Source Field**—Field on which to filter the information
- **Row Source**—Row source to use in the filter
- **Row Source Column**—Row source column to use in the filter
- **Match**—Select **True** to allow only matching records in the filter. Select **False** to allow only non-matching records in the filter.
- **Non-matching are Errors**—Whether to log data values that do not match with filter values in the error log file. Select **True** or **False**. (The default is **False**, data values that do not match filter values are not entered in the log file.)

- **Dimension Filter**

- **Hierarchy Level**—Hierarchy level on which to filter the information
- **Data Source Field**—Field on which to filter information
- **Dimension**—Dimension to use in the filter
- **Filter On**—Select Namespace or Hierarchy. If you select Namespace, enter a value. (If you leave the Namespace value blank, the default hierarchy is used.) If you select Hierarchy, select a hierarchy and a hierarchy level.

See [Dimension Namespaces](#) and [Dimension Hierarchies](#).

- **Allow Members**—Select **True** to include only the records that exactly match the dimension member names. Select **False** to include only the records that do not match dimension member names.
- **Non-matching are Errors**—Whether to log data values that do not match with filter values in the error log file. Select **True** or **False**. (The default is **False**, data values that do not match filter values are not entered in the log file.)

- **Distinct Filter**
 - Hierarchy Level—Hierarchy level on which to filter information
 - Filter Value—Value for the filter. To include columns in the filter value, click **Add** and select a column.
 - Distinct—Whether data should have distinct values across the data set. Select **True** or **False**.
 - Non-matching are Errors—Whether to log data values that do not match filter values in the error log file. Select **True** or **False**. (The default is False, data values that do not match filter values are not entered in the log file.)
- **Custom Filter**
 - Hierarchy Level—Hierarchy level on which to filter information
 - Class Name—Java class name that contains the custom filter logic
 - Parameters—Parameters passed to the Java class

5 Define Custom Processing information.

- **Custom Pre-processing**

To define custom preprocessing details, click **Create** and enter:

- **Class Name**—Java class containing pre-processing logic before staging starts for this stage map
- **Parameters**—Parameters passed to the pre-processing Java class

To define parameters, click **Add** and enter a value, type, and format for the parameter.

- **Custom Post-processing**

To define custom pre-processing details, click **Create** and enter:

- **Class Name**—Java class containing postprocessing logic after staging ends for this stage map
- **Parameters**—Parameters passed to postprocessing Java class

To define parameters, click **Add**, and enter a value, type, and format for the parameter.

6 Click **Save**

Creating and Editing Allocation Maps

➤ To create and edit allocation maps:

- 1 On the Object Browser **View** menu, select **Cubes** and select a cube, and then click **Actions** and select **Edit**.
- 2 Click **Allocation maps**.
- 3 To create an allocation map, click **Create**. To edit a map, click an allocation map and click **Edit**.

4 Enter **Properties** information.

- Name
- Source Dimension

5 Define the **Allocation Scope**.

In the Source Scope and in the Target Scope, for each dimension, enter or review:

- Scope Type—For example: not mapped, all, level, levels, nonLeaves
- Value—For example, if you selected levels for the Scope Type, then you could select Summary, Plant, Department or Critical Resource.

The functions for the Source Scope Members and Target Scope Members are shown on the Cube Wizard, under **Allocation map details**:

Source Scope Members example:

```
[level(ProductLine.ProductLine."Product Line")] [levels(Resource.Resource.Summary, Resource.Resource.Plant, Resource.Resource.Department)]
```

Target Scope Members example:

```
[level(ProductLine.ProductLine."Product Line")] [levels(Resource.Resource.Plant, Resource.Resource.Department, Resource.Resource."Critical Resource")]
```

6 Enter or review **Dimension Map** information for the source and target dimension.

To view detailed information for a dimension, select the dimension name. Information is displayed under Dimension Map Details. To change dimension map information, click **Edit**.

- Dimension
- Namespace—Name to uniquely identify dimension members

See [Dimension Namespaces](#).

- Mapping Type—Select Simple or Fixed Member

For Simple mappings, define the type of mapping and the dimension to which to map.

See [“Mapping Functions” on page 194](#).

For Fixed Member mappings, select the member.

7 Click **Save** .

Reviewing and Publishing Model Objects

Model objects include data sources, row sources, dimensions, cubes, and stagemaps. When you publish an object, it is saved as a base object in Integrated Operational Planning,

► To review and publish changes made to objects in the Model:

- 1 In the **Administration Workbench**, go to the **Model** tab and click **Review Changes**.
- 2 Review **Model Changes**.

Model changes include data sources, row sources, dimensions, cubes, and stagemaps that have been added, edited, or deleted since the last time objects were published.

3 Review and fix **Validation Errors**.

Integrated Operational Planning validates all Model objects. Objects for which validation fails are highlighted. The cause of the error is explained under Validation Errors.

You cannot publish objects until all validation errors are fixed.

4 On the **Publish** page, select a script template.

The script templates that are displayed are defined on the Script Templates page in the Administration Workbench. See [“Managing Script Templates” on page 133](#). Types of script templates include:

- **Initial Publish**—Publishes the model for the first time
- **Re-Publish**—Republishes model changes

For example, select Re-Publish when you add measures to a cube, update a measure, add a cube, add a mapping, or make similar model-level metadata changes

- **Re-Publish with Load**—Republishes model changes *and* data changes

For example, select Re-Publish with Load when you add a data source along with the source data.

5 Click **Publish**.

6 Click **Done**.

➤ To undo a model change before it is published:

- 1 In the **Administration Workbench**, go to the **Model** tab and click **Review Changes**.
- 2 Select the check box next to the desired model change.
- 3 Click **Revert**.

Changing Models After Publishing

Subtopics

- [Exporting Models](#)
- [Importing Models](#)

You make changes after publishing for the following reasons:

- For Model related importing, see [“Changing Models After Publishing” on page 109](#)
- For workbook, worksheet, queries related to importing

You make changes after publishing for the following reasons:

- Model related importing

- For workbook, worksheet, queries related to importing, see [“Changing Models After Publishing” on page 109](#).

After you publish models there are restrictions:

- You cannot delete an object or the relationship between objects.
- You cannot add a new dimension to a published cube.
- You cannot delete an existing dimension from a published cube.
- You cannot change the row source column type.
- You cannot delete a constraint.

To work around these restrictions, administrators can use the export and import functionality to make changes to the model objects:

➤ To make changes to models after publishing:

- 1 **Export model objects.**
- 2 **Reset and restart the system:**
 - a. Stop the server and reset the system.
 - b. Run `isreset`.
 - c. Run `startserver`.
 - d. In a new window, run `initializesystem -u <admin privileged username>-p <password>`.
- 3 **Import the model.**
- 4 **Make the necessary changes.**
- 5 **Publish the model.** See [“Reviewing and Publishing Model Objects” on page 108](#).

Exporting Models

➤ To export models:

- 1 In the **Administration Workbench**, in the Model tab, click **Review Changes**, then click **Export/Import**, and then click **Export**.
- 2 The following objects are exported:
 - Documentation
 - Scenarios
 - User Filters
 - Model Definitions
 - User Preferences
 - Database Connection Definitions
- 3 Run `isreset`.

- 4 Restart the server.

Importing Models

➤ To import models:

- 1 In the **Administration Workbench**, in the **Model** tab, click **Review Changes**, and then click the **Export/Import** tab.
- 2 Click **Import**.

Using the Presentation Tab

Subtopics

- [Managing Analysis Types](#)
- [Managing Queries](#)
- [Managing Report Templates](#)
- [Reviewing and Publishing Model Objects](#)
- [Changing Models After Publishing](#)
- [Managing Workbooks](#)
- [Managing Worksheets](#)

Managing Analysis Types

Subtopics

- [About Analysis Types](#)
- [Creating Analysis Types](#)
- [Adding Key Metrics to Analysis Types](#)
- [Adding Key Assumptions to Analysis Types](#)
- [Viewing and Editing Analysis Type Details](#)
- [Deleting Analysis Types](#)

About Analysis Types

Analysis types define a planning workbook, one or more report workbooks, key metrics, key assumptions, participants, and approvers.

For analysis types that include a data collection workbook, you can assign default data providers and define the e-mail template used to send them instructions.

The analysis types defined in the Presentation tab are displayed in the Planning Workbench as you create scenarios. Each scenario is associated with an analysis type.

See [“Working with Scenarios” on page 29](#).

Creating Analysis Types

► To create analysis types:

1 In the **Administration Workbench**, go to the **Presentation** tab and select **Analysis Types**.

2 Click **Add** and enter analysis type details:

- **Type**—Select an option:
 - Analysis—Used for what-if analysis
 - Data Collection—Used in data collection scenarios. After selecting, scroll to **Data Collection Settings**:
 - Data collection workbook—Enter the order of the workbook
 - Default data providers—Select **Users/Groups** or **Query**
 - Default e-mail form—

In data collection scenarios, data providers receive data collection workbooks to update forecast data. The e-mail form is used in data collection scenarios to e-mail instructions and an attached data collection workbook.
- **Name**—Identifies the analysis type in Integrated Operational Planning
- **Display Name**—(Optional) Name to display in the Planning Workbench if different than the name already defined. Use a name that helps identify the purpose of the analysis type. For example, for data collection, add “input” or “import” to the name.

Description—Analysis type description. For example, a description for an SOP Analysis could be: “Allows customers to align supply and demand plans.”

3 Associate a **Planning Workbook** and one or more **Report Workbooks** with the analysis type.

Planning workbooks are used for what-if analysis in scenarios associated with the analysis type. Report workbooks are displayed in the Reports section of the Planning Workbench for the analysis type.

View details for the available workbooks in the Workbooks section of the Presentation.


4 Select **Assigned Constraints**.

Only selected constraints are then visible to the user on his profile. See [“User Profile” on page 28](#).

5 Select default **Participants**.

Participants can review scenarios with the analysis type and provide comments. Analysis owners can remove default participants and add others when they create a scenario.

To select participants, do one of the following:


- Select **Users/Groups** and click  to select participants from the available users and groups.
- Select **Query** and define an SQL query to select the participants.

6 Select additional **Approvers**.

Approvers can review scenarios with the analysis type, provide comments, and approve plan changes.

Analysis owners can add approvers when they create a scenario. Analysis owners cannot remove Approvers.

To select additional approvers, perform an action:

- Select **Users/Groups** and click  to select approvers from the available users and groups.
- Select Query and define an SQL query to select approvers.

7 Click **OK**.

The new analysis type is displayed in the Presentation tab and appears as an option when you create a scenario in the Planning Workbench.

Adding Key Metrics to Analysis Types

Key metrics can be added only to saved analysis types. If you are creating an analysis type, you must save it before you can add key metrics.

► To add key metrics to an analysis type:

1 In the **Administration Workbench**, go to the **Presentation** tab and select **Analysis Types**.

2 Select an analysis type.

3 In **Key Metrics**, click **Add** and enter the following information:

- **Name**
 - Assign a name that is close to the dimension member name (from the cube's measure dimension); for example, *Projected Sales*.
 - Indicate whether the key metric value is absolute or relative; for example, if the value is relative, you could add *Delta* to the name, as in *Projected Sales Delta*.
 - Indicate the data type of the key metric; for example, whether the value is in dollars (as in *Projected Sales \$*) or units (*Projected Units*).
- **Description**
- **Users**: If no users are assigned, the key metric has no restrictions and is assigned to all users. If a key metric needs user restrictions, then add the users assigned to use the key metric.
- **Type**—Whether the key metric is comprised of members or a query definition. (Select a **Style** for the type you select.)
 - **Members**—Select a cube and define one member from each dimension in the cube, including the measure dimension.

To define a member, click Add and enter the fully qualified name of a dimension member in the form *dimension name / namespace / member name*. The measure dimension and its namespace have the same name as the cube.

For example, consider a key metric, named `Projected Sales Delta`, corresponding to a measure named `Projected Sales`, which is contained in the `Forecast` cube. The `Forecast` cube includes the dimensions `ProductLine`, `Geography`, `Customer`, `Fiscal`, and `Forecast`. (The `Forecast` dimension is the measure dimension.)

For example, to define the key metric `Projected Sales Delta` as projected sales for all customers in all regions for all product lines for Q4 of 2004, you would add the following members:

`ProductLine/ProductLine/ProductLine`

`Geography/Geography/Geography`

`Customer/Customer/Customer`

`Fiscal/Fiscal/2004 Q4`

`Forecast/Forecast/Projected Sales`

If you cannot save a key metric definition, ensure that all members are from the selected cube and one member is selected from each dimension in the cube. (View cube details on the `Model` tab in the `Administration Workbench`.)

See [“Creating a Cube” on page 93](#).

- **Query Definition**—Enter an MDX query to define the key metric.

Sample query definition for gross margin:

```
select {SystemPeriod([Fiscal].[Year])} ON COLUMNS
FROM [Financials]
WHERE ([Measures].[Gross Margin %])
```

Click **Preview** to display the results of the query definition.

4 Click **OK**.

Adding Key Assumptions to Analysis Types

Key assumptions can be added only to saved analysis types. If you are creating an analysis type, you must save it before you can add key assumptions. Examples of key assumptions include inflation, currency exchange rates, and labor rates.

► To add key assumptions to an analysis type:

- 1 In the `Administration Workbench`, go to the **Presentation** and select **Analysis Types**.
- 2 Select an analysis type.
- 3 In **Key Assumptions**, click **Add** and enter the following information:
 - Name
 - Description
 - Sheet Name—Report sheet where key assumption data can be uploaded

The sheet names that appear are the report worksheets defined on the Worksheets page in the Presentation tab. See [“Creating Worksheets” on page 123](#).

- **Style**—Number format style applied in the Show Impact window when analyzing the scenario
- **RSQL Query Definition**—Required to select the key assumption value from the row source

Sample RSQL query definition for an exchange rate:

```
select EXCHANGE_RATE from Currency_Exchange_RS where TERM_CURRENCY='EUR'
```

Click **Preview** to display the results of the query definition.

4 Click .

Key assumptions are displayed in the Key Metrics/Assumptions section when you select a scenario from the Analysis Workbench. They are also displayed in the Show Impact window in Excel. When you click an assumption value in the Key Impact window, the report sheet defined when creating the assumption (see Sheet Name in [step 3](#) above) is opened.

Viewing and Editing Analysis Type Details

➤ To view or edit analysis types:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Analysis Types**.
- 2 Select an analysis type to view or edit its details.

Deleting Analysis Types

You cannot delete an analysis type that is being used by a scenario. The scenario must first be deleted (in the In Progress or Complete state) or completed and then deleted (in the Submitted or Approved state).

➤ To delete analysis types:

- 1 In the **Administration Workbench**, go to the **Presentation** and select **Analysis Types**.
- 2 Select an analysis type.
- 3 Click **Delete**.

Managing Queries

Subtopics

- [Understanding Queries](#)
- [Creating Queries](#)
- [Previewing and Editing Queries](#)
- [Deleting Queries](#)

Understanding Queries

Types of queries in Integrated Operational Planning:

- **Custom**—Uses custom Java classes.
- **MDX (Multidimensional Expressions)**—Uses standard MDX and Integrated Operational Planning extensions to query multidimensional cubes. MDX queries return subsets of the base data or scenario data in the Integrated Operational Planning database.
- **Overlay**—Uses Integrated Operational Planning tools in Excel to create a planning worksheet layout. The layout sets the level to which a dimension hierarchy or list of measures is expanded or collapsed and focuses the view to display specific measures and dimension members.
- **RSQL (Row Source Structured Query Language)**—Uses SQL to return row source data. Row source data is transactional, component-level data (for example, sales order details) loaded into relational tables in the Integrated Operational Planning database from an execution database.

Select an option from the menu at the top right of the Queries screen to define the type of queries to display.

Creating Queries

► To create queries:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Queries**.
- 2 Click **Add** and enter query details:
 - **Name**—Must be one word
 - **Type**—Custom, MDX, Overlay, or RSQL
 - **Owner**
 - **Description**
 - **Max Rows**
 - **Execution Class Name**—For custom queries, the Java class name used to execute the query

- **Definition**—You can copy and paste the query definition from another source, such as a text file.

Syntax requirements:

- A comma is required after an `ON COLUMNS` statement only if it is followed by an `ON ROWS` statement.
- You can place line breaks between keywords; however, do not place line breaks inside dimension member names.
- A `Parameter(...)` statement is required in an MDX query for each filter menu in a report worksheet.

3 Click **Preview** to confirm query results and correct syntax errors.

You can preview the query against base data or scenario data.

See [“Previewing and Editing Queries” on page 117](#).

4 Click **OK**.

Previewing and Editing Queries

You preview a query to confirm that it returns the expected results. When you preview a query, a syntax check is performed and errors are reported.

For new report worksheets, the number of columns required in the report is the same as the number of columns in the preview table. The number of columns depends on the hierarchy level of the dimension members specified in the query definition.

You can preview a query while creating it, and you can preview an existing query. You can preview the query by using base data in Integrated Operational Planning or by using scenario data.

➤ To preview or edit a query:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Queries**.
- 2 To search for a query, enter text in the text boxes above the columns and click **Enter**. To cancel the query, clear the text boxes and click **Enter**.
- 3 In the **Name** column, select a query.
- 4 Click **Preview** and select whether to preview the query against base data in Integrated Operational Planning or against scenario data.

If you preview the query using scenario data, select a scenario from the menu. If no names are displayed, no scenarios exist in the In Progress or Submitted state.

A separate browser window opens to display the query definition and syntax errors.

5 Correct errors in the query definition, and then click **Preview** to confirm your corrections.

Each time you preview the query, a browser preview window opens. If no errors exist, the query runs and a table of results is displayed.

Considerations when previewing a query:

- Depending on the query, it may take time to retrieve results from the Integrated Operational Planning database. Keep the preview window open until the query finishes running.
- MDX queries pause during preview whenever a `Parameter (. . .)` statement is encountered in the query definition. A dialog box opens, prompting you to enter a value for the parameter. Enter a value and click OK.

For example, if the parameter represents a dimension member name, enter the fully qualified dimension member name in the form *dimension name/namespace name/member name*; for example, *Manufacturing/MY 2004/August*.

- RSQL queries pause during preview each time the variable “?” is encountered in the query definition. A dialog box opens, prompting you to enter a value to substitute for the “?” variable. Enter a value and click OK.

Deleting Queries

► To delete queries:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Queries**.
- 2 Select a query.
- 3 Click **Delete**.

Managing Report Templates

Subtopics

- [Adding Report Templates](#)
- [Viewing and Editing Report Template Details](#)
- [Viewing and Editing Report Templates in Excel](#)
- [Deleting Report Templates](#)

Report templates are Excel templates used when creating new reports. The following section describe how to manage report templates.

Adding Report Templates

► To add a report template:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Report Templates**.
- 2 Click **Add** and enter report template details:
 - **Name**—Report template name
 - **Description**—Report template description
 - **Owner**—Report template owner

- **Visibility**—Private or Public:
 - **Private**—Can be viewed and used only by the template owner and the administrator.
 - **Public**—Can be viewed and used by everyone.
- **File name**—Excel file name; for example, Simple-Report-FixedHeader.xls


Viewing and Editing Report Template Details

➤ To view or edit report template details:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Report Templates**.
- 2 Select a report template to view or edit its details.

Viewing and Editing Report Templates in Excel

➤ To view or edit a report template in Excel:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Report Templates**.
- 2 Click  next to the report template name.

Deleting Report Templates

➤ To delete a report template:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Report Templates**.
- 2 Select a template.
- 3 Click **Delete**.

Reviewing and Publishing Model Objects

Model objects include analysis types, worksheets, workbooks, and report templates. When you publish an object, it is converted to a base object in Integrated Operational Planning.

➤ To review and publish changes made to objects in the Presentation tab:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and click **Review Changes**.
- 2 **Review Model Changes.**

Model changes include analysis types, worksheets, workbooks, and report templates that have been added, edited, or deleted since the last time objects were published.

- 3 **Review and fix Validation Errors.**

Integrated Operational Planning validates all Presentation objects. Objects for which validation fails are highlighted. The cause of the error is explained under Validation Errors.

You cannot publish objects until all validation errors are fixed.

4 Click **Publish**.

5 Click **Done**.

► To undo a model change before it is published:

1 In the **Administration Workbench**, go to the **Presentation** tab and click **Review Changes**.

2 Select a model change.

3 Click **Revert**.

Changing Models After Publishing

Subtopics

- [Exporting Models](#)
- [Importing Models](#)

To change models after publishing in the Model tab, see [“Changing Models After Publishing” on page 109](#).

Exporting Models

► To export models:

1 In the **Administration Workbench**, in the **Presentation** tab, click **Review Changes**, then click **Export/Import**, and then click **Export**.

2 Run `isreset`.

3 Restart the server.

Importing Models

► To import models:

1 In the **Administration Workbench**, in the **Presentation** tab, click **Review Changes**, and then click the **Export/Import** tab.

2 Click **Import**.

Managing Workbooks

Subtopics

- [Understanding Workbooks](#)
- [Creating Workbooks](#)
- [Viewing and Editing Workbook Details](#)
- [Deleting Workbooks](#)

Understanding Workbooks

Report, data collection, and planning workbooks, together called analysis workbooks, are “virtual” workbooks assembled from worksheets stored in Excel workbook files on the Integrated Operational Planning server. A workbook must contain at least one worksheet. The worksheets associated with an analysis workbook can be the same as those in an Excel workbook file, but need not be.

Workbook descriptions:

- **Report Workbooks**—Contain only report worksheets. Report workbooks are associated with an analysis type and are listed in the Reports section in the Planning Workbench. Analysis owners, participants, and approvers use report workbooks to view tabular reports and charts.
- **Data Collection Workbooks**—Contain at least one report worksheet designed to collect updated forecast data from data providers. Data collection workbooks can contain other report worksheets, but they should not contain planning worksheets. Data collection workbooks are associated with an analysis type that is used to create a data collection scenario.
- **Planning Workbooks**—Contain at least one planning worksheet and, optionally, report worksheets. Planning workbooks are associated with an analysis type. Analysis owners use the analysis type to create scenarios, and analysis owners and participants use a scenario planning workbook to perform what-if analyses.

Creating Workbooks

➤ To create a workbook:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Workbooks**.
- 2 Click **Add** and enter workbook details:
 - **Name**
 - **Type**—Report, Data Collection, or Planning
 - **Description**—Workbook description
 - **Run against**—Select an option:
 - Any—Run the report against any data

- BaseOnly—Run the report only against base data
- ScenarioOnly—Run the report only against a scenario

3 Associate worksheets with the workbook.

Planning workbooks can contain planning and report worksheets. Data Collection workbooks can contain data collection and report worksheets. Report workbooks can contain only report worksheets. Available worksheets are defined in the Worksheets section of the Presentation tab.

4 Click **OK**.

Viewing and Editing Workbook Details

► To view or edit a workbook:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Workbooks**.
- 2 Select a workbook to view or edit its details.

Deleting Workbooks

► To delete a workbook:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Workbooks**.
- 2 Select a workbook.
- 3 Click **Delete**.

Note: You cannot delete planning, data collection, or report workbooks associated with an analysis type. You must edit the analysis type first to remove the association.

Managing Worksheets

Subtopics

- [Creating Worksheets](#)
- [Viewing Worksheet Details](#)
- [Hiding Worksheets](#)
- [Editing Worksheets](#)
- [Deleting Worksheets](#)

Understanding Worksheets

Types of worksheets in Integrated Operational Planning:

- **Report Worksheets**—Provide access to Integrated Operational Planning class modules that call Integrated Operational Planning queries. To use a report worksheet, create a query that retrieves and filters data from the Integrated Operational Planning database. You then use

design tools and the Visual Basic Editor in Excel to create the report layout and specify the query to use.

- **Data Collection Worksheets**—Report worksheets that allow data providers to update forecast data. These worksheets reference a pre-existing load specification that defines how updated forecast data is staged and loaded into the Integrated Operational Planning database. The worksheets include information used to filter and display forecast data for individual data providers.
- **Planning Worksheets**—Created by customizing XML worksheet template files outside Integrated Operational Planning.

Creating Worksheets

➤ To create a report or data collection worksheet:

1 In the **Administration Workbench**, go to the **Presentation** tab and select **Worksheets**.

2 Click **Create** and select **Report Worksheet** or **Data Collection Worksheet**.

3 Enter worksheet details:

- **Type**—Report or Data Collection
- **Description**
- **Load Specification**—Path to a loader XML file that defines how to load the data collection XLS sheet into a row source or table. The loader XML file should reside in / install/custom/loader.

If you create a report using a predefined report library template (see [“Adding Report Templates” on page 118](#)), you can download the library template from your client computer, make any desired changes, and use the XML file defined here to upload the changes.

- **Owner**—Worksheet owner
- **Permission**—Private or Public

Only the worksheet owner and administrator can view private worksheets.. All users can view public worksheets.

- **File name**—For example, mfg-analysis.xls
- **Sheet name**—Name of the Excel worksheet

4 **Optional:** Define required worksheets.

The Required Sheets option is displayed only when you set `reportsheet.includedependents=true` in properties files.

5 Click **OK**.

➤ To create a planning worksheet:

1 In the **Administration Workbench**, go to the **Presentation** tab and select **Worksheets**.

2 Click **Create**, and select **Planning Worksheet**.

Excel opens to display a new workbook containing a blank worksheet.

3 Click , or select **Oracle**, and then **Worksheet Designer**.

4 In **Worksheet Designer**, perform the following actions:

- a. Click **Select**, and enter an Excel data range.
- b. Select a cube.
- c. Click **Create Default** to create a default layout using the selected cube (the default layout includes all dimensions, their attributes, and a default measure), or click **Start** to manually create a layout for this worksheet.

5 Review or edit the information in the **Worksheet Designer** tabs:

- **Model**—Data range, anchor cell, and cube details

To redesign the sheet, click **Revert**. (Clicking **Revert** deletes only the Worksheet Designer data; it does not delete Excel data.)

- **Dimensions**—All dimensions and their sheet locations

Available functions:

- **Unmap Dimension**—Unmaps the dimension and creates an “unknown” dimension
- **Map Dimension**—Maps an “unknown” dimension to an available dimension
- **Map Range**—Maps a dimension to a range
- **Map Attribute**—Maps a dimension attribute to a data range
- **Unmap Attribute**—Unmaps a dimension attribute

Use **Unmap Attribute** if you are adding another string and need to remap the attribute.

- **Delete Attribute**—Deletes a dimension attribute

Use **Delete Attribute** if you do not want all the columns in the default.

- **Measures**—All measures and their sheet locations

Available functions:

- **Unmap Measure**—Unmaps the measure and creates an “unknown” measure
- **Map Measure**—Maps an “unknown” measure to an available measure

- **Publish**—Defines the worksheet name and description, whether the worksheet is hidden, and whether the worksheet is private.

6 In the **Publish** tab, click **Save**.

Note that this action does not publish the worksheet. See [“Reviewing and Publishing Model Objects” on page 119](#) for information on how to publish Presentation objects (including worksheets).

Viewing Worksheet Details

As an administrator, you can view details for planning, report, and data collection worksheets that are registered in Integrated Operational Planning. Registered worksheets are available for use in planning, report, and data collection workbooks.

Each worksheet belongs to an Excel workbook. The file name of a registered worksheet is given by the file name of the Excel workbook that contains it. An Excel workbook can contain other worksheets.

► To view details for registered planning, report, and data collection worksheets:

1 In the **Administration Workbench**, go to the **Presentation** tab and select **Worksheets**.

2 Review worksheet details:

- **Excel Sheet**—Name of the Excel worksheet. Select a name to open the worksheet in Excel.
- **Type**—Planning, Data Collection, or Report
- **Accessible to**—Who can access the worksheet
- **Excel File**—Excel file name
- **Published**—Whether the worksheet has been published

See [“Reviewing and Publishing Model Objects” on page 108](#).

- **Modified**—Whether the worksheet has been modified since it was created.
 - **Private**—Whether the worksheet is public or private. Only the worksheet owner and the administrator can view private worksheets. All users can view public worksheets.
 - **Hidden**—Whether the worksheet is hidden
- See [“Hiding Worksheets” on page 125](#).
- **Dependents**—Whether the worksheet is dependent on another worksheet. For example, a report worksheet may depend on data in a hidden worksheet. Or a Refresh Data button in one worksheet may also refresh data in another worksheet.

In Excel, you cannot insert a dependent worksheet into a planning workbook. On the Presentation tab, you can add a dependent worksheet to a workbook, but you must also add the worksheets on which it depends. Otherwise, problems may occur when the workbook is used in scenarios.

- **Description**—Worksheet description

Tip: Click a column header to sort the list of worksheets.

Hiding Worksheets

You can hide worksheets in a workbook. For example, you could hide a worksheet used to retrieve data from the Integrated Operational Planning database for use in a chart on a different worksheet. In Excel, hidden report worksheets are invisible to all users, including administrators.

➤ To hide a worksheet:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Worksheets**.
- 2 Select a worksheet.
- 3 Click **Visibility**, and select **Hide**.


Note: Hidden worksheets are not the same as private worksheets. Hidden worksheets are invisible to all users and are used to avoid displaying data that is not directly useful to users. Private worksheets restrict the view of data to prevent access by other users.

Editing Worksheets

➤ To edit a report or data collection worksheet:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Worksheets**.
- 2 Select a worksheet to edit its details.

➤ To edit a planning worksheet:

- 1 In the **Administration Workbench**, go to the **Presentation** tab and select **Worksheets**.
- 2 Select a worksheet to open it in Excel.
- 3 Click , or select **Oracle**, and then **Worksheet Designer**.
- 4 In the **Worksheet Designer**, edit the worksheet details, then select the **Publish** tab, and then click **Save**.

Deleting Worksheets

➤ To delete a worksheet:

- 1 In the **Administration Workbench**, select the **Presentation** tab, and then select **Worksheets**.
- 2 Select a worksheet.
- 3 Click **Delete**.

Using the Administration Tab

Subtopics

- [Managing Essbase Connections](#)
- [Managing the Job Queue](#)
- [Managing the Script Editor](#)
- [Managing Script Templates](#)
- [Managing Security Filters](#)
- [Managing System and Application Resources](#)
- [Managing Users and Groups](#)

Managing Essbase Connections

Subtopics

- [Adding Essbase Connections](#)
- [Deleting Essbase Connections](#)

The Essbase Connections tab displays all Essbase connections (including inactive connections) and their status. You can use these connections to create data sources in the Model.

Adding Essbase Connections

➤ To add an Essbase connection:

- 1 In the **Administration Workbench**, go to **Administration** and select **Essbase Connections**.
- 2 Click **Add**.
- 3 Enter connection details.
 - **Name**—Oracle Essbase connection name
 - **Description**—Essbase connection description
 - **Essbase Instance**—Instance where Essbase is installed
 - **Host**—Essbase host name
 - **Username**—User name used to set up the Essbase connection
 - **Password**—Password for the Essbase connection
 - **Application Name**—Essbase application name

Click **Populate Applications** to load all the applications in the specified Essbase Instance into the Application Name list.

 - **Database name**—Essbase database name
- 4 Click **Test** to test the connection.
- 5 Click **OK**.

Note: Database connections other than Essbase are managed through the Oracle WebLogic Server Administration Console.

Deleting Essbase Connections

► To delete an Essbase connection:

- 1 In the **Administration Workbench**, go to **Administration** and select **Essbase Connections**.
- 2 Select a connection.
- 3 Click **Delete**.

You cannot delete a connection to an Oracle Essbase application that is running.

Managing the Job Queue

Subtopics

- [About the Job Queue](#)
- [Adding Jobs](#)
- [Viewing Jobs](#)
- [Editing Job Details](#)
- [Executing Jobs](#)
- [Deleting Jobs](#)

About the Job Queue

Use the job queue to execute and monitor the status of data staging and loading operations and dimension restructuring.

The job queue displays details for jobs initiated manually and for jobs executed using batch scripts.

Batch scripts can be used to accomplish these tasks:

- Update data in the Integrated Operational Planning database
- Restructure dimensions and recalculate cube data if the underlying metadata has been modified
- Delete unnecessary jobs from the job queue; for example, when multiple entries for the same job occur

Tip: Data loading jobs are generally scheduled and executed using batch scripts.

Note: Using the job queue to manually execute large jobs can consume system resources and slow browser response.

Adding Jobs

► To add a job to the job queue:

- 1 In the **Administration Workbench**, go to **Administration** and select **Job Queue**.
- 2 Click **Add** and enter job details:
 - **Type**—Job category (used as the display name in the job queue)
 - **Description**
 - **Owner**
 - **Error Fatal**—Whether errors during job execution will interrupt the job
 - **Command**—Commands executed by the Integrated Operational Planning server to perform the job
- 3 Click **OK**.

Note: You can use the Integrated Operational Planning `isadmin` command-line tool to add a custom job to the job queue. Oracle recommends that you do not edit `isadmin` commands for predefined jobs.

See [Appendix A, “Isadmin Commands.”](#)

Viewing Jobs

► To view job details:

- 1 In the **Administration Workbench**, go to **Administration** and select **Job Queue**.
- 2 At the top of the Job Queue, select either **Open Jobs** or **Executed Jobs**.
 - Open jobs are not executed or are currently running. They are in Pending or Active states.
 - Executed jobs are completed or were attempted. They are in Completed, Completed with Warnings, or Failed states.
- 3 Review job details:
 - **Type**—Job category (for example, restructuring)
 - **Creation Date**—For open jobs, the date and time the job was added to the job queue
 - **Status**—Job status:
 - **Pending**—Not invoked
 - **Active**—Running
 - **Completed**—Completed successfully
 - **Completed with Warnings**—Completed with errors that did not interrupt job execution
 - **Failed**—Failed with errors that interrupted job execution

- **Owner**—

The owner of a data collection scenario is automatically the owner of data collection jobs such as data staging and data loading.

- **Error Fatal**—Whether errors during job execution will interrupt the job

- **Command**—Commands executed by the Integrated Operational Planning server to perform the job

- **Description**

4 Click a job to view additional job details, including the complete list of commands to perform the job.

Executed jobs include the following additional information:

- **Execution Date**—Date and time the job was executed (completed or attempted)

- **Execution Detail**—Name and location of an error log file

Editing Job Details

► To edit job details:

1 In the **Administration Workbench**, go to **Administration** and select **Job Queue**.

2 Select a job to edit its details.

Note: You can edit jobs only in the Pending state.

Executing Jobs

You can manually execute jobs in the Pending state. Jobs are executed sequentially in the order they were created. If you execute one job of a certain type, all jobs of that type in the job queue are executed.

► To manually execute a job:

1 In the **Administration Workbench**, go to **Administration** and select **Job Queue**.

2 Select the desired job.

3 Click **Execute**.

Note: Jobs run in the background. Depending on the job, your browser response may be slow during job execution.

Note: You can monitor job progress in the DOS window where the Integrated Operational Planning server is running. You can view job history in the console log file `isserver.log` `interlace_root/logs`, where `interlace_root` is the Integrated Operational Planning installation directory.

Deleting Jobs

► To delete a job in the job queue:

- 1 In the **Administration Workbench**, go to **Administration** and select **Job Queue**.
- 2 Select the analysis type.
- 3 Click **Delete**.

Note: You cannot delete a job with an Active status.

Managing the Script Editor

Subtopics

- [Understanding Script Files](#)
- [Adding Script Files](#)
- [Viewing Script Files](#)
- [Deleting Script Files](#)

Understanding Script Files

Script files contain commands to execute a workflow. Integrated Operational Planning uses script files to build models and to configure business processes. For example, JACL files can be used to invoke JAVA classes on server startup, and JAVA script files can be used to customize a model or reports.

Adding Script Files

► To add a script file:

- 1 In the **Administration Workbench**, go to **Administration** and select **Script Editor**.
- 2 Click **Add** and enter script file details:

- **Name**
- **Type**—JACL Script or Java Script

See [“Script Templates and JACL Scripts”](#) on page 158.

- **Filename**—Any valid name containing alphanumeric characters (no spaces)

When you create a new file, you must select a file type to use as the file extension. For example, a file named *dbutil_1* with a file type of *jacl* will be saved on the Integrated Operational Planning server as *dbutil_1.jacl*. Script files are stored in `/install/custom/scripting`.

- 3 Enter script contents.

Script contents vary depending on the type of file. Sample JACL script:

```

#
# Common db utilities
#
# $Id: //interlace/projects/properties/base/main/resource/interlace/jacl/
dbutils.jacl#1 $
#

proc getTableCount { connection tablename } {

    set sql "SELECT COUNT(*) AS CNT FROM $tablename"
    set cstmt [$connection createStatement]
    set rs [$cstmt executeQuery $sql]
    set count -1;
    if [$rs next] {
        set count [$rs getInt "CNT"]
    }
    $rs close
    $cstmt close
    return $count;
}

proc checkTableExists { connection tablename } {

    set sql "SELECT 1 FROM $tablename"
    set cstmt [$connection createStatement]

    set count 0
    java::try {
        set rs [$cstmt executeQuery $sql]
        $rs close
        set count 1
    } catch {SQLException e} {
        set count 0
    }

    $cstmt close
    return $count;
}

```

4 Click **OK**.

Viewing Script Files

► To view script files:

- 1 In the **Administration Workbench**, go to **Administration** and select **Script Editor**.
- 2 From the **Filter** menu, select the type of script files to view (All, JAACL script, or Java script).

Deleting Script Files

► To delete a script file:

- 1 In the **Administration Workbench**, go to **Administration** and select **Script Editor**.

- 2 Select the script file.
- 3 Click **Delete**.

Managing Script Templates

Subtopics

- [Understanding Script Templates](#)
- [Adding Script Templates](#)
- [Editing Script Templates](#)
- [Deleting Script Templates](#)

Understanding Script Templates

Script templates are used to publish model changes in Integrated Operational Planning. You can create both system and non-system script templates.

- **System script templates** are available when publishing objects in the Model Review Changes Wizard.

See [“Reviewing and Publishing Model Objects” on page 108](#).

Types of system script templates:

- Initial Publish—Publishes the model for the first time
- Re-publish—Republishes model changes
- Re-publish with Load—Republishes model changes *and* data changes
- **Non-system script templates** along with access rights can be assigned to users to execute in the Planning Workbench. These templates are displayed under Scripts on the Planning Workbench Home page.

See [“Working with Scripts” on page 71](#).

Adding Script Templates

➤ To add a script template:

- 1 In the **Administration Workbench**, go to **Administration** and select **Script Templates**.
- 2 Click **Add** and enter template details:
 - **Name**—Script template name
 - **Description**—Purpose of the script template
 - **Contents**—Commands to execute a predefined workflow (for example, publishing a model)
 - **System**—Whether the template is a system script template

See [“Understanding Script Templates” on page 133](#).

- 3 Click **OK**.

Editing Script Templates

- To edit a script template:
- 1 In the **Administration Workbench**, go to **Administration** and select **Script Templates**.
 - 2 Click on a script template name and change the template details:
 - **Name**
 - **Description**
 - **Contents**—Commands to execute a predefined workflow (for example, publishing a model)
 - **System**—Whether the template is a system script templateSee [“Understanding Script Templates” on page 133](#).
 - 3 Click **OK**.

Deleting Script Templates

- To delete a script template:
- 1 In the **Administration Workbench**, go to **Administration** and select **Script Templates**.
 - 2 Select a script template.
 - 3 Click **Delete**.

Managing Security Filters

Subtopics

- [Understanding Security Filters](#)
- [Adding Security Filters](#)
- [Editing Security Filters](#)
- [Deleting Security Filters](#)

Understanding Security Filters

Security filters set access levels on cube data (dimension and measure members), thereby preventing users from updating or deleting specific sections of data.

Adding Security Filters

► To add a security filter:


1 In the **Administration Workbench**, go to **Administration** and select **Security Filters**.

2 Click **Add** and enter security filter details:

- **Name**
- **Access Level**—No Access, Read Only
 - By default, users have full privileges on the system
 - To change the default level on a set of cells, the administrator uses the MDX query
 - The administrator can change the default to no access or read-only by selecting the access level
 - If there is a conflict on a data security level on a cell, the more stringent security is used
- **Query**—MDX query name

The queries displayed are the MDX queries defined on the Queries tab in the Presentation tab on the Queries tab.
- **Description**—Security filter description

3 Assign users to the security filter.

- a. Click .
- b. **Optional.** Search for a specific user by entering search criteria at the top of the Select Users dialog box.
- c. Select users.
- d. Click **Add** to add selected users, or click **Add All** to add all users.
- e. Click **OK** to exit from the Select Users dialog box.

4 Click **OK** to add the security filter to Integrated Operational Planning.

Editing Security Filters


► To edit a security filter:

1 In the **Administration Workbench**, go to **Administration** and select **Security Filters**.

2 Click a security filter name and edit details:

- **Access Level**—No Access, Read Only
- **Query**—MDX query name

The queries displayed are the MDX queries defined on the Queries tab in the Presentation tab on the Queries tab.
- **Description**—Security filter description

- 3 Assign users to the security filter.
 - a. Click .
 - b. **Optional:** Search for a user by entering search criteria at the top of the Select Users dialog box.
 - c. Select users.
 - d. Click **Add** to add selected users, or click **Add All** to add all users.
 - e. Click **OK** to exit from the Select Users dialog box.
- 4 Click **OK** to save the changes to the security filter to Integrated Operational Planning.

Deleting Security Filters

- To delete a security filter:
- 1 In the **Administration Workbench**, go to **Administration** and select **Security Filters**.
 - 2 Select a security filter.
 - 3 Click **Delete**.

Managing System and Application Resources

- To manage system and application resources:
- 1 In the **Administration Workbench**, go to **Administration** and select **System**.
 - 2 In the **System Tools** section, select **Display System Logs**.
 - a. In the **Log Types** section, select a log type.
 - b. To open a file or download a file to the client, in System Logs Files section, select a log file.
 - 3 Review **Spreadsheet Properties**:
 - **max.rows**—Maximum rows a zoom or a search can display
 - **max.columns**—Maximum columns a zoom or a search can display
 - **display.options.max.formula.length**—Maximum formula characters to show in a cell comment
 - 4 Review **Error Logs**.

Select an error log to view its details.

Managing Users and Groups

Subtopics

- [Adding Users and Groups](#)
- [Assigning Access Privileges](#)
- [Viewing Users and Groups](#)
- [Searching for Users and Groups](#)

Manage users and groups in Shared Services. View users and groups in Integrated Operational Planning.

Adding Users and Groups

➤ To add, modify, or delete users or groups from Integrated Operational Planning:

- 1 From **Administration Workbench**, select the **Administration** tab, and then select **Users and Groups**.
- 2 Click the **Shared Services Console** link.

Note: To add, modify, or delete users and group, you must use Shared Services. The Users and Groups tab in the Administration Workbench is for viewing users and groups only.

Assigning Access Privileges

➤ To assign access privileges to a user or group:

- 1 From **Administration Workbench**, click the **Administration** tab, and then click **Users and Groups**. To select a user or group, click a **Login Name**.

The Edit User or Edit Group screen is displayed.

- 2 In **Provide Object Access** section, in **Select Object Type**, select **Analysis Types**, **Report Workbooks**, or **Script Templates**.

The objects for the selected object type are displayed.

- The analysis types displayed are defined in the Analysis Types tab in the Model Designer.
 - The report workbooks displayed are defined in Workbooks tab in the Model Designer.
 - The script templates displayed are defined in the Script Templates tab under Administration. Only nonsystem script templates are displayed (the System field on the Script Templates tab is set to false).
- 3 Select an object and an access option.
 - For Analysis Types, you can provide access to *create/update/delete* the analysis type.
 - For Report Workbooks, you can provide access to *read* the report workbook.
 - For Script Templates, you can provided access to *execute* the script template.

- 4 Click **OK** to save the object access assignments.

Viewing Users and Groups

- To view users and groups:

- 1 From **Administration Workbench**, click the **Administration** tab, and then click **Users and Groups**.
- 2 The following user information is displayed:
 - **Login Name**—User login name
 - **Provider Name**—Provider name of the user; for example, Native Directory
 - **Full Name**—Used as the display name
 - **E-mail Address**—Address where notification messages are sent
 - **Logged in**—Whether the user is currently logged in to Integrated Operational Planning

The following group information is displayed:

- **Name**
 - **Provider Name**
 - **Description**
- To toggle between users and groups, click **Users** or **Groups**.

Searching for Users and Groups

- To search for users and groups:
- 1 From **Administration Workbench**, click the **Administration** tab, and then select **Users and Groups**.
 - 2 Enter search criteria (**User Property**, **User Filter**, **In Groups**), and click **Search**.



Isadmin Commands

In This Appendix

About Isadmin Commands.....	139
Isadmin Commands and XML Files.....	142
Using Isadmin Commands in Integrated Operational Planning.....	143
Isadmin Commands	145
Scripts in install-root/bin.....	164
Process Flow Modeling	164
Evolution of the Commands Through Releases	165
Scripts.....	165

About Isadmin Commands

Subtopics

- [Command Options](#)
- [Common Activities](#)
- [Common Usage](#)

Isadmin is the client-side command-line tool that interacts with the Integrated Operational Planning application server.

Command Options

Table 5 isadmin Command Options

Option	Description
-f	File containing statements to execute
-h	Help
-l	Log file
-p	Password
-r	URL (default = <code>http://hostname:port/interlace</code> , where <i>hostname</i> and <i>port</i> are set in <i>machine_name</i> . properties)

Option	Description
-s	Statement to execute
-u	User name
-v	Server name (default = localhost)

Common Activities

Common activities of `isadmin` commands include:

- **Archive**—Archives data from the current period before moving to the next period. Archived data is used for waterfall and other reports. An example is shown in the sample model in *install-root/samples/sample*. See the JACL script, `archives.jacl` for details. Integrated Operational Planning uses a snapshot query to output the data and store it in an archive table.
- **Clear values**—Clears user-input values and returns to the original calculated or loaded value. See the weekly script in the sample model in *install-root/samples/sample* for details.
- **Snapshot**—Exports a slice of the cube using an MDX query. See the archive script in the sample model in *install-root/samples/sample* for details.
- **Waterfall reports**—Compares a set of data as time moves forward. For example, a waterfall report could compare the forecast for the current quarter and for future quarters. See the weekly script in the sample model in *install-root/samples/sample* for details.
- **ETL**—Extract-transform-load. Sometimes the data feed is not formatted to go directly into the system. In Integrated Operational Planning, the entry point is the data source. Before pointing the data source to an XLS file, a CSV file, or a table, Integrated Operational Planning uses JACL scripts to do SQL manipulation. See the JACL script, `archives.jacl` for patterns of SQL activities.
- **Send e-mail**—See the weekly and daily scripts in *install-root/samples/sample* for examples of sending e-mail notifications. These scripts are entry-level scripts invoked from the client. Sometimes, Integrated Operational Planning must send e-mails from the server (for example, in the middle of the execution of script template or JACL scripts). See [“JACL Scripts” on page 165](#).
- **Single thread script execution**—Prevents concurrent execution of scripts. For example, if you accidentally invoke a daily script twice, you do not want the restructured code to run in two separate threads; instead, you can set exclusive execution of scripts and exclusive access to the shared resources in JACL scripts. See the weekly and daily scripts in the sample model in *install-root/samples/sample* for details.

Common Usage

- Execute the statement “export model definitions” and define the user name and password to connect to the server.

```
isadmin -u admin -p password -s "export model definitions"
```

- Execute the statements in `commands.isa` and define the username and password to connect to the server.

```
isadmin -u admin -p password -f "c:\osop\custom\bin\load.isa"
```

The `isa` extension is commonly used as the extension for files containing `isadmin` commands. You must define the full path for the file.

- Execute multiple commands interactively by creating a “shell” and entering commands one at a time.

```
C:\dev>isadmin -u admin -p password
Logged in as admin
Admin> create sandbox new_sandbox
executing: [create sandbox new_sandbox]
Admin> alter sandbox set batch mode
executing: [alter sandbox set batch mode]
Admin> alter sandbox calculate
executing: [alter sandbox calculate]
Admin> submit sandbox
executing: [submit sandbox]
Admin> logout
executing: [logout]
Syntax error at line 1 column 1
logout
^
Admin> exit
C:\dev>
```

This creates a session in the server and all commands are executed within that session. In the previous example:

- Users create a shell by entering the command as shown.
- The client responds by displaying the prompt `Admin`.
- Users enter the commands one at a time.
Commands are displayed in blue, and client responses are displayed in black.
- Each command makes a round trip to the server, and users can monitor the server log. Syntax errors are highlighted.
- Users complete the session by entering `exit`.

Note: In Windows, you can access previous commands by using the up and down arrow keys.

Isadmin Commands and XML Files

Subtopics

- [XML Schema Files](#)
- [XML Files and Search Path](#)

XML Schema Files

Metadata objects, scenario objects, and other data objects imported and exported from Integrated Operational Planning conform to the grammar specified in the XML schema files in:

install-root/etc/schema

You can use the Administration Workbench to generate objects. If XSD functionality is not supported, use XML to generate the objects.

XML Files and Search Path

When referring to XML files in an `isadmin` command, the system searches the following directories:

install-root/custom/directory-representing-object-type
install-root/interlace/directory-representing-object-type

Table 6 Search Path for Objects in the System

Object Type	Directory Searched
Analysis types	<i>install-root/custom/analysis</i>
Cube, Dimension, Constraints	<i>install-root/custom/model</i> <i>install-root/interlace/model</i>
Custom property files	<i>install_root/custom/config</i>
Data source	<i>install-root/custom/datasource</i>
Exported information	<i>install-root/export</i>
ISA files and other OS scripts	<i>install-root/custom/bin</i>
JACL scripts	<i>install-root/custom/jacl</i>
Report library templates	<i>install-root/interlace/reportLibrary</i>
Row source	<i>install-root/custom/rowsource</i>
Script templates	<i>install-root/custom/scripts</i>
SQL scripts	<i>install-root/custom/sql</i>
Stagemap and loader files	<i>install-root/custom/loader</i>

Object Type	Directory Searched
User-defined custom scripts (JACL, JS)	<i>install-root/custom/scripting</i>
Workbook, worksheets, named queries, styles, and XLS files used in worksheets	<i>install-root/custom/workbook</i>

Using Isadmin Commands in Integrated Operational Planning

➤ To use `isadmin` commands in Integrated Operational Planning:

- 1 Install Integrated Operational Planning, set up the database, modify the properties, and set up `PATH` and other environment variables.

See the *Integrated Operational Planning Installation Guide* for details.

- 2 Ensure that the server is not running; then, in a DOS window, enter `isreset`.

`Isreset` drops all internal tables and model information, and creates the internal tables again. There is no model at this time.

- 3 Enter `startserver` to start the server.

The server starts and listens on the configured port.

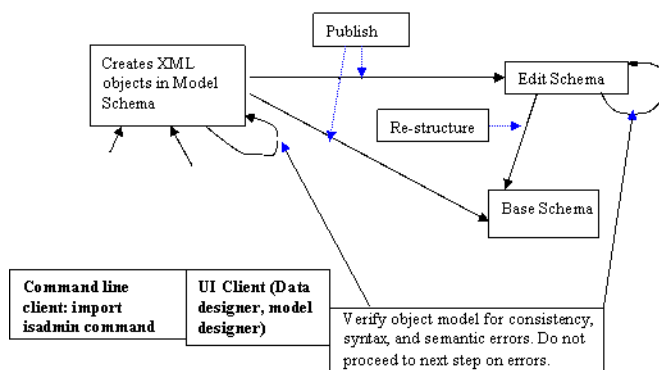
- 4 Open another DOS window and enter `initializesystem.bat`.

This action initializes the objects needed to build a model.

- 5 Use the Administration Workbench to build a model.

Figure 11 shows the flow that happens in the system.

Figure 11 Flow for Building a Model



The objects created in the Administration Workbench are stored in the Model schema. You can verify errors, fix issues, and publish the objects for analysis. Integrated Operational Planning stores the models in the database, where they are available between server starts.

- 6 **Optional.** Export model information to the file system.

Reasons to export model information:

- Moving the implementation from one computer to another (for example, from development to a test or production computer)
- Backing up the model in the file system
- Accessing functionalities that are only supported in XML
- Accessing model objects in XML form when writing scripts

To export model information to the file system, use the `isadmin` command `export model definitions`. The objects are exported to *install-root/export* and various sub-directories.

See [“XML Files and Search Path” on page 142](#).

Integrated Operational Planning also exports an `isa` file that contains import statements for modeling objects in the system. The objects and `isa` file reset the system and use scripts to recreate the model.

7 Copy the subdirectories under *install-root/export* to *install-root/custom*.

The export directory is used only for storing the exported files. You can make incremental changes in the custom directory and write scripts to import the changes back to the system. After the model is imported back to the system, use the Administration Workbench to verify the model and make more changes if needed.

8 Repeat incremental model building by following [step 5](#), [step 6](#), and [step 7](#).

Isadmin Commands

Subtopics

- [Macro](#)
- [Schema](#)
- [Import](#)
- [Publish](#)
- [Loading Members into Dimensions and Building Hierarchies](#)
- [Export](#)
- [Generate](#)
- [Load](#)
- [Sandbox Calculation, Submit, and Reconcile](#)
- [Script Templates and JACL Scripts](#)
- [System](#)
- [Export MultiDimensional Data](#)
- [Custom Java Code Commands](#)
- [System Utility](#)

Macro

Macro commands are used to avoid modifications in the script every time an object is added. Because macro commands are resolved dynamically by the system at runtime, script maintenance is simplified. As the object model changes, the macro commands automatically adapt to the changes.

For example, you can use a macro command to publish all *object_type* definitions. The syntax to do this is as follows:

```
publish object_type definitions.
```

where *object_type* can be a data source, row source, dimension, stagemap, cube, or constraint.

Using row source as an object type, the following example publishes all row source definitions. The semantics of “all” is determined at runtime. If you add an object type, you do not need to change the script.

```
publish rowsource definitions
```

Other commands that can be used as a macro command include:

- `load {replace|update}`
- `stage {replace|update|clear}`
- `export model definitions`
- `export all scenarios`

Schema

Subtopics

- [alter model schema set current edit](#)
- [set schema shadow](#)
- [clear schema shadow](#)

Schema commands involve defining the state of the Integrated Operational Planning schema.

alter model schema set current edit

Function

Sets the model schema to edit mode.

Syntax

```
alter model schema set current edit
```

Arguments

None

Example

```
alter model schema set current edit
```

set schema shadow

Function

Sets the schema to shadow.

Syntax

```
set schema shadow
```

Arguments

None

Example

```
set schema shadow
```

clear schema shadow

Function

Clears any previous information.

Syntax

```
clear schema shadow
```

Arguments

None

Examples

```
clear schema shadow
```

Import

Import shadowed metadata objects such as cubes and dimensions; non-shadowed metadata objects such as worksheets and named queries; scenarios and scenario-specific information into Integrated Operational Planning.

Syntax

- `import datasource|rowsource|dimension|stagemap|cube|constraint|reporttemplate|modelgroup|dbconnection|worksheet|workbook|analysistype|style|scope|rule} {definition|definitions} from file xml_file_name`

Arguments

xml_file_name—XML file containing the object definitions

- `import scenario from directory file_name {replace using batch}`

Arguments

file_name—Name of the directory containing the scenario definitions. The scenario from the specified directory is imported.

- `import {named queries|acls|security filters} from file xml_file_name`

Arguments

xml_file_name—XML file containing the object definitions. Imports the respective named queries, acls and security filters from the xml file specified.

- `import cubemaps`

Creates cube maps.

- `import {user preferences|filters}`

Imports user preferences from export/preferences directory and filters from export/filters directory.

Description

The object definitions are in the XML file referred to in the command, and the XML file conforms to the grammar specified by the Integrated Operational Planning XSD. See [“XML Files and Search Path” on page 142](#) for information on how to locate the XML file.

On a clean system, the first action is to import these objects. If an object is already in the system and you run import again, Integrated Operational Planning compares the imported object with the existing object to see if anything has changed.

Examples

- The following Imports the data source, row source, dimension, stagemap, cube, constraint, and reporttemplate definitions:

```
import datasource definition from file "${interlace_home}/custom/model/
datasource_IN_TRANSITTIME_DS.xml"
import rowsource definition from file "${interlace_home}/custom/model/
rowsource_IN_SCRAPPERCENT_RS.xml"
import dimension definition from file "${interlace_home}/custom/model/
dim_Fiscal.xml"
import stagemap definition from file "${interlace_home}/custom/model/
stagemap_IN_TRANSITTIME_SM.xml"
import cube definition from file "${interlace_home}/custom/model/
cube_DriveDemand.xml"
import constraint definition from file "${interlace_home}/custom/model/
constr_HDAExcess.xml"
import reporttemplate definition from file "${interlace_home}/custom/reportlibrary/
report_Simple Report(Fixedheader).xml"
```

- Imports database definition:

```
import all dbconnection definition [from file file name]
```

Imports all database connection information: The command imports database connection object details to an XML file in install-root/custom/datasource. The default file name is dbconnections.xml.

- The XML file is generated by the corresponding export command
- The XML file contains all the parameters that defines the connection to the Essbase instance
- To verify the parameters, click **Administrative Workbench**, then **Administration**, and then **Essbase connections**.

Publish

Checks for objects imported into the system and publish the objects so that they are available to everyone.

Syntax

```
publish {datasource|rowsource|dimension|stagemap|cube|constraint}  
definitions
```

or

```
publish {datasource|rowsource|dimension|stagemap|cube|constraint|  
dbconnection worksheet|workbook|reporttemplate|keymetric|analysistype}  
definition object_name
```

Arguments

object_name—Name of the object to publish.

Description

When publishing, Integrated Operational Planning checks to ensure that the object model is valid.

Publish commands refer to the following schemas supported by Integrated Operational Planning.

- **Model schema**—Schema into which XML objects are imported and maintained. XML objects are created through actions in the Administration Workbench and by `isadmin` import commands.

XML files that do not conform to XSD are rejected. Integrated Operational Planning does additional syntax and semantic checking not expressible in XSD to ensure a consistent and valid object model.

Model objects include *shadowed objects* (cubes, dimensions) and *nonshadowed objects* (workbooks and worksheets used for presentations). When publishing, shadowed objects go first to the Edit schema and then to the Base schema. Nonshadowed objects go directly to the Base schema.

- **Base schema**—Schema into which XML objects that are converted to proprietary backend versions of objects are maintained. This is the published schema available to everyone. All loading, calculations, and data manipulation occur in this schema.
- **Edit schema**—Similar to the Base schema for storing backend versions of objects. Integrated Operational Planning creates the backend versions of XML objects in the Edit schema, does more error checking not possible in the Model schema, and then gets ready to push the changes to the Base schema.

Examples

The following example publishes row source, data source, dimension, and stagemap definitions.

```
# Set the model schema to edit mode.  
  
alter model schema set current edit  
  
# Set the schema to shadow in backend and clear any previous information.  
  
set schema shadow
```

```

clear schema shadow

# Publish row source and dimension definitions to shadow schema.
# These "macro" commands publish the imported data sources, row sources, dimensions, and
# stagemaps.
# This way you don't need to remember to publish one object at a time.

publish rowsource definitions
publish datasource definitions
publish dimension definitions
publish stagemap definitions

# Push object definitions from shadow schema to base schema.

alter system restructure

```

The following example publishes cube and constraint objects. Publish cubes *after* you publish and restructure dimensions.

```

# Set the model schema to edit mode.

alter model schema set current edit

# Set the schema to shadow in backend and clear any previous information.

set schema shadow
clear schema shadow

# Publish cube and constraint definitions to shadow schema.
# These "macro" commands publish the imported cube and constraint definitions.
# This way you don't need to remember to publish one object at a time.

publish cube definitions
publish constraint definitions

# Push object definitions from shadow schema to base schema.
alter system restructure

```

Loading Members into Dimensions and Building Hierarchies

Subtopics

- [stage replace](#)
- [load replace dimensions](#)

Loading members into a dimension is a metadata load. During member load, dimensional structures such as members, member attributes, hierarchies, and levels are created and populated. (This is metadata information.) As a result, it is important to do a member load in the shadow schema and then restructure to publish changes to the base schema.

The members of a dimension and how they relate to each other are important when creating the multidimensional space within which the system operates.

The following example loads members into dimensions and restructures the hierarchy:

```
# Set the schema to shadow in backend and clear any previous information.

set schema shadow
clear schema shadow

# Load dimensions and restructure so dimensions and members are
# in the base schema. The macro replace command replaces all dimensions.
# The reference to model group name is from the stagemap definitions.
# You can logically group the stagemaps using this name
# and the macro command automatically invokes all the stagemaps
# grouped under this name. You can create
# different logical groups, put the stagemaps under different
# groups, and invoke them separately.

stage replace using modelgroup "Bootstrap-Dimension-Stagemaps"
load replace dimensions

# Push object definitions from shadow schema to base schema.

alter system restructure
```

stage replace

Function

Loads dimensions and restructures so that dimensions and members are in the base schema.

Syntax

```
stage replace using modelgroup stagemap_group_name
```

Arguments

stagemap_group_name—The reference to the model group name from the stagemap definition. You can logically group stagemaps using this name. The macro command automatically invokes all stagemaps grouped under this name.

Description

Use `stage replace` *before* `load replace dimensions`. The stagemaps in the group name specified are considered for loading. You can create different logical groups, put the stagemaps under different groups, and invoke them separately

Example

```
stage replace using modelgroup "Bootstrap-Dimension-Stagemaps"
```

load replace dimensions

Function

Replaces all dimensions.

Syntax

```
load replace dimensions
```

Arguments

None

Description

Use `load replace dimensions` *after* `stage replace`. Use `stage` for the stagemap group, load the dimensions, and restructure to apply the changes to the base schema.

Example

```
load replace dimensions
```

Export

Description

Export commands export shadowed metadata objects such as cubes and dimensions, nonshadowed objects such as worksheets and named queries, and scenarios and scenario-specific information out of Integrated Operational Planning.

Syntax

- `export model definitions to file_name`
Exports all model objects.
- `export {dimension|cube|constraint|datasource|rowsource|stagemap|dbconnection|worksheet|workbook|reporttemplate|style|keymetric|analysistype|rule|scenario|modelgroup|data|documentation} to file file_name`
Exports specific objects.
- `export all scenarios`
Exports all scenarios.
- `export {scenario|modelgroup|test scenario|data|documentation} to file file_name`
Exports the scenario, modelgroup, data or documents to the specified XML file
- `export all dbconnection definition to file_name`
Export all dbconnection objects.
- `Export user preferences|filters|acIs`
Exports all user preferences, filters or acIs generating the respective xml files.

Arguments

file_name—XML file name in which to export the object

Description

Export commands export shadowed metadata objects such as cubes and dimensions, nonshadowed objects such as worksheets and named queries, and scenarios and scenario-specific information out of Integrated Operational Planning.

Examples

- `export model definitions to ${interlace_home}/custom]`
Exports all metadata to `${interlace_home}/custom`. The different object types go into their respective directories; for example, data source to `install-root/custom/datasource`.
- `export all scenarios`
Exports all scenarios to `${interlace_home}/export/scenarios`. Each scenario is exported to a new directory.
- `export documentation to file ${interlace_home}/custom/doc/model.html`
Exports the documentation of the object model in the system. The command exports object details and their relationship as a HTML file in `install-root/custom/doc`. The default file name is `model.html`.
- `export all dbconnection definition [to file file name]`
Exports all database connection information in the system. The command exports database connection object details to an XML file in `install-root/custom/datasource`. The default file name is `dbconnections.xml`.

Generate

The Generate function applies a set expression or a string expression to each member of a specified set.

Syntax

- `generate forecast forecast_file_name using generator generator_name`

Description

Generates the forecast data file with the specified name containing the data generated using forecast pro tool. The generator name is the class name which contains the database query to get the required input data.

Arguments

- `forecast_file_name`—Name of the forecast file generated
- `generator_name`—Name of the forecast generator used

Example

```
generate baselineforecast using generator BaselineForecast
```

- generate model statistics

Description

Generates information about model statistics that include cube and row sources as well as the system statistics. This will be useful for modelers to understand the size of their model. The generated information is stored as a file `model_sizing_guide.html` under the directory `install_root/custom/doc`.

Arguments

none

Example

```
Generate model statistics
```

Load

Subtopics

- [stage](#)
- [load](#)
- [Troubleshooting Load Commands](#)

Load commands load external data into the system. Integrated Operational Planning loads data into a new sandbox, thereby not affecting the base sandbox. Upon successful load and calculation, `isadmin` commands can submit the changes to the base sandbox.

Load commands include:

- [stage](#)
- [load](#)

Each step can specify the semantics of the load in the corresponding command. Integrated Operational Planning supports these semantics:

- **Update**—For each input row, compose the record using the target's key columns, and search for that record in the target. If the key exists, replace all non-key values in the target row with values from the input row. If the key does not exist, add the input row.
- **Replace**—Delete all target rows and add all input rows.
- **Append**—Insert the input row at the end of the target.
- **Clear**—Clear the stage tables. The combination of clear and update commands are same as the replace command.

Integrated Operational Planning uses *update* when the input data stream is incremental; for example, inventory at the end of the current week. *Replace* is used when the data stream is a complete snapshot of the data in the external system.

Tip: Always do a “stage replace” even for incremental data streams. The load update of row source incrementally updates the row source.

Tip: Staging is not done inside a sandbox, and the stage tables are common resources. As a result, stage commands are normally placed outside the sandbox.

The following commands deal with incremental data streams. The commands do a stage replace followed by a load into row sources and cubes.

```
# The model group concept is similar to that of dimensions. The
# stagemaps for row sources are also grouped using different names.
# It is important to match this name with that in the stagemap.
# Otherwise, skip those row sources.stage replace using modelgroup "Bootstrap-Rowsource-
Stagemaps"
# create sandbox, load all row sources, calculate and submit sandbox
# The macro definitions allow us simplify the commands and the system
# resolves the names of all row sources to be loaded.
# When you create sandbox in the following command, you set a new sandbox to be the
current sandbox in current session and is applicable to all
# subsequent commands. When sandbox is submitted, the current sandbox
# gets reset to base sandbox.
```

```
create sandbox "data_bootstrap"
alter sandbox set batch mode
load update rowsources
alter sandbox calculate
submit sandbox
```

The following commands stage and replace a single row source:

```
stage replace using stagemap IN_MEDIA_METADATA_SM"
create sandbox "media_meta_sb"
alter sandbox set batch mode"
load replace rowsource IN_MEDIA_METADATA_RS"
alter sandbox calculate
submit sandbox
```

The following command clears the stage tables of row sources. Any subsequent update is like a replace:

```
stage clear rowsources
```

stage

Description

Stages data from a data source into stage tables.

Syntax

stage {update|replace|append} from {file|datasource|database} *file_name* using {file|stagemap|stagemaps} *xml_file_name*

stage {update|replace|append} using {stagemap|stagemaps} *stagemap_names*

stage clear {rowsource|cube|dimension} *object_name*

stage clear {rowsources|bom|dimensions}

Arguments

- *object_name*—Name of the object; for example, row source, cube, or dimension name
- *xml_file_name*—XML file name used to stage the data. This can be a loader xml file with load specifications or a stagemap file.
- *file_name*—File containing the staged data; for example, a CSV or text file containing the input data
- *stagemap_names*—A single stagemap name or a list of stagemap names

Example

```
stage update using stagemap \"SM_IntercompanyDemandRise\"
stage update from file \"$f\" using file \"$migrationHome/loader/load-rowsource-$rs.xml\"
stage update from datasource Orders using stagemaps OrdersMap,OrderLinesMap
stage update from datasource Forecast using stagemap ForecastMap
stage replace using stagemap \"ForecastHistory\"
stage replace from file \"demo-bom.csv\" using file \"load-bom.xml\"
stage replace using stagemaps \"ETLSalesOrders_SalesOrders\" ,
\"ETLSalesOrders_DailySalesOrders\"
```

load

Description

Loads data from the stage tables to a row source.

Syntax

```
load {update|replace} {rowsource|dimension|cube} object_name
load {update|replace} {rowsources|dimensions|bom|users}
load datasources using {dimension|rowsource} stagemaps identified by group group_name
load clear rowsource object_name
```

Arguments

object_name—Name of the object; for example, row source name or dimension name.

Example

```
load update users
load update cube \"Schedule\"
```

```
load update rowsource ForecastHistory
load replace rowsource \"SalesForecast\"
load update rowsource \"SuggestedGrossMargin\"
load replace dimensions
load replace bom
load replace rowsources
```

Troubleshooting Load Commands

To troubleshoot Load commands, see [“Load Commands” on page 286](#).

Sandbox Calculation, Submit, and Reconcile

Subtopics

- [create sandbox](#)
- [alter sandbox](#)
- [submit sandbox](#)

Sandbox calculation, submit, and reconcile commands load seed data into the blocks of the cube through row source loading and the mapping between the row source and different cubes. These values appear in loaded measures. In addition, other measures in the cube must be calculated.

create sandbox

Function

Create a sandbox.

Syntax

```
create sandbox name
```

Arguments

name—Name of the sandbox.

Example

```
create sandbox data_calculate
```

alter sandbox

Function

Alters a previously-created sandbox.

Syntax

```
alter sandbox {set batch mode|calculate|restructure}
```

Arguments

None

Example

```
Alter sandbox set batch mode  
Alter sandbox calculate  
Alter sandbox restructure
```

submit sandbox

Function

Submits the current sandbox.

Syntax

```
submit sandbox
```

Arguments

None

Example

```
submit sandbox
```

Script Templates and JACL Scripts

Script templates and JACL scripts provide a convenient way to group commands. The commands can be executed from the file.

Script templates are similar to isa files in that they are group of commands; however, they are also class objects managed inside the system. You can use the Administration Workbench to create and modify script templates, associate the permissions of who can run them, and execute them from the user interface.

The following command imports script template information from the file system.

```
import script template from file "RegenerateAll.xml"
```

The XML file conforms to the XSD in *install-root/etc*.

The following command executes a script template and all the isadmin commands inside the script template.

```
alter system invoke script template "Initial Publish"
```

The name used in the command is the name defined inside the script template object (or XML file).

JACL scripts intermix `isadmin` commands with other activities such as polling external data source, sending e-mails, extracting data, calling APIs, and integrating with other systems. When the server starts, it prepares the script execution environment, and it is ready to accept any JACL scripts.

Note: See [“Perl Module Scripts” on page 167](#) for how to write the JACL script made available by the server during execution.

Use the [invoke](#) command to invoke the external JACL script.

invoke

Description

Invokes the external JACL script.

Syntax

```
invoke external script file_name parameters
```

Invoke external script using file `ignoreMissing` name. If `ignoreMissing` is specified, the exception is ignored.

Arguments

- *file_name*—JACL script file name
- *name*—Name of the script to be executed
- *parameters*—Parameters required to execute the script. Use spaces to separate the parameters.

Example

The following examples use JACL scripts; however, they are applicable to other types of scripts as well.

- ```
invoke external script "load-mdpothers.jacl"
```

Invokes the JACL script.
- ```
invoke external script "load-mdpothers.jacl" "daily" "c:/osop/  
install/ftproot"
```

Invokes the JACL script with two arguments.

Example

The following property lists the directories searched by the system to locate a script file:

```
loader.upload.script.directories
```

To use the directories listed in the property to locate the script (JACL) file, enter the following code:

```
invoke external script "upload_data.jacl"
```

The default, specified in `ISServer.properties` is set to:

```
loader.upload.script.directories=${interlace.home}/custom/scripting,$
${interlace.home}/custom/workbook,$${interlace.home}/custom/jacl,$
${interlace.home}/custom/scripts,$${interlace.home}/interlace/workbook
```

If you specify a fully-qualified name, then the property is not relevant. For example, in the command below, the system simply searches the file:

```
invoke external script "c:\iop\custom\upload_dir\upload_data.jacl"
```

System

System commands perform system maintenance activities.

alter system

Function

Perform maintenance activities.

Syntax

```
alter system {gc|javagc|allow adminonly|allow all|prepare for migration|
populate reporting schema|reconcile active response|reconcile response
number|set datetime to datetime force|set maintenance mode|unset
datetime|unset maintenance mode|freeze datetime at datetime|invoke
script template script_name|invoke application hook|execute tasks in
group group_name|add task group group_name command command user
user_name onfailure boolean_token|delete tasks in group group_name|
reset|restructure|solve constraint using rule rule rule_name|enable
email reader|disable email reader|purge all closed scenarios|purge
closed scenarios older than number days}
```

Arguments

- *cube_name*—Name of the cube
- *number*—Any valid number
- *datetime*—Any valid date/time
- *script name*—Script template name
- *command*—Command to execute
- *user_name*—Name of the user

- *boolean_token*—True or false
- *group_name*—Model group name
- *rule_name*—Constraint rule name

Description

The `alter system` command includes the following maintenance activities:

- **Garbage collection**—Forces the Java server to run garbage collection at the end of large batch calculations or at the end of a script.
- **Prepare for migration**—Exports all metadata and data to the file system, including the data in the row sources and the user-entered data in the cubes. Derived data is not exported.
- **Migration**—Brings in the information from exported files and uses `isadmin` commands to import the model back into the system. The following command imports user-entered values in cube cells back into the cubes.

```
load update cube "DriveDemand"
```

This command looks for *cube-name-XXXX.csv* files in directory *export-dir/migration/data*, where *XXXX* is a counter and it starts from 0000. Cube data is exported to *export-dir/migration/data* with a maximum of 50000 rows to a file.

See [“Migration Scripts” on page 168](#) for a sample migration script.

- **Populate reporting schema**—Integrates Integrated Operational Planning with Oracle Business Intelligence Enterprise Edition. The command creates view definitions on OIOP schema tables. The data is used by Oracle Business Intelligence Enterprise Edition to generate reports and dashboards.
- **Set/unset datetime**—Sets/unsets the user provided date.
- **Set/unset maintenance mode**—In administrative mode, you are not allowed logins to the user interface, even by an administrator. Set the system to single-user mode during large data load and calculations, and to normal mode at the end of the script. Disable e-mail at the beginning of a script and enable it again at the end.
- **Freeze datetime**—Freezes the application date/time.
- **Invoke script template**—Invokes the script template.
- **Execute tasks**—Executes the tasks.
- **Add task**—Adds a task in the group.
- **Delete tasks**—Deletes tasks from the application.
- **Restructure**—Pushes the object definitions from the shadow schema to the base schema.
- **Solve constraint**—Uses [Custom Java Code Commands](#) to solve constraints.
- **Enable/disable e-mail reader**—Enables/disables the e-mail reader. Disable the e-mail reader while running large calculations.
- **Purge closed scenarios**—Reduces the clutter in the user interface and improves performance.

Examples

```
alter system gc
```

```
alter system purge all closed scenarios  
alter system purge closed scenarios older than 30 days
```

```
alter system set maintenance mode  
alter system disable e-mail reader
```

Export MultiDimensional Data

Integrated Operational Planning supports MDX queries that retrieve a slice of cubular data. Using `isadmin` commands, you can export this same data to relational tables to be used elsewhere. You can also use these commands to archive information on a recurring basis before a new data refresh.

The following command takes a query name as an argument, creates a relational table, and populates the table with data returned by the query.

```
snapshot data using query queryname (append|replace)
```

queryname refers to the named query in the system. You must create this query before running the command. `append` adds the data to the end of the table. `replace` drops all the rows and then adds the new rows to the table.

Custom Java Code Commands

Custom Java Code commands are a set of “standard” Java classes that are configurable in the `properties` file. With these standard Java classes, you do not need to write any Java code—you just need to understand what the command does and the parameters it takes. Enter the following to invoke these commands:

```
alter system solve constraint using rule "MyRule1"
```

`MyRule1` is defined in the `properties` file as follows:

```
ConstraintSolver.MyRule1.classname=com.interlacesystems.rules.PreviousValueCopier  
ConstraintSolver.MyRule1.description=Copies current week values to previous week  
ConstraintSolver.MyRule1.SourceMeasures=MS DEMAND  
ConstraintSolver.MyRule1.TargetMeasures=LAST WEEK MS DEMAND  
ConstraintSolver.MyRule1.Cube=DriveDemand  
ConstraintSolver.MyRule1.Condition=isNotPast  
ConstraintSolver.MyRule1.Levels=DriveMedia-Level1,Week
```

Custom Java classes are *solvers*: they solve a violation of some business rule or constraint. Standard solvers take cube, levels, and condition as arguments.

Solver behavior includes:

- **Copy previous value**—Copies values from one measure to another using the classname `com.interlacesystems.rules.PreviousValueCopier`. Preserves the values from one week (month) to another (month), and is invoked before loading values for the next

time period. The new value is stored in the source measure after the data refresh, values between the two time periods are compared.

- **Clear user input**—Clears user-input values and returns to the original calculated or loaded value. The following code retrieves baseline loaded values

```
ConstraintSolver.MyRule.classname=com.interlacesystems.rules.ClearUserInput
ConstraintSolver.MyRule.description=Clear Measures from DriveDemand cube
ConstraintSolver.MyRule.Cube=DriveDemand
ConstraintSolver.MyRule.Measures=MS DEMAND,MS DRIVE DEMAND
```

In this example, the classname is `com.interlacesystems.rules.ClearUserInput` and the parameters are `cube` and `measures`.

Conditions supported by Integrated Operational Planning:

- `isCurrent`—Only the current time members of the cube participate in the solver work. As time elapses, the solver works on different time members.
- `isPast`—Only the past time members participate in the solver work.
- `isFuture`—Only the future time members participate in the solver work.
- `isNotPast`—The current and future time members participate in the solver work.
- `isNotCurrent`—The past and future time members participate in the solver work.
- `isNotFuture`—The past and current time members participate in the solver work.

System Utility

System Utility commands are used to auto generate scripts that help build models.

Auto Generate Loader Specification

Description

Generates a loader specification XML file to load Excel data from the specified Excel worksheet to a database table.

The generated loader XML file is placed in `<install-root>/custom/loader` with a file name of `load-<sheet-name>.xml`. The table drop and create JACL script is generated in `<install-root>/custom/loader/scripting` with a file name of `Loadscript-<sheet-name>.jacl`. The input Excel file should be placed in `<install-root>/custom/data`.

Syntax

```
generate loadspec for sheet [<sheetName>] from file [<xls fileName>]
```

Arguments

- *SheetName*—Excel worksheet from which to load data
- *xls fileName*—Name of the XLS file from which the *SheetName* is taken

Scripts in install-root/bin

The following scripts are in *install-root/bin* and help manage the system:

- `anteros.bat`
- `anteros_install.bat`
- `anteros_uninstall.bat`
- `converter.bat`
- `createiopinstance.bat`
- `generateanterosreg.bat`
- `initializesystem.bat`
- `isadmin.bat`
- `isreset.bat`
- `pingserver.bat`
- `pingurl.bat`
- `setenv.bat`
- `setmaintenancemode.bat`
- `showversion.bat`
- `unsetmaintenancemode.bat`

Process Flow Modeling

The core functionality supported in Integrated Operational Planning is multidimensional model building, planning, and analysis. It is important to consider:

- The processes that manipulate the system
- The business stakeholders associated with these processes
- The frequency of these processes
- What permissions to associate with these processes
- What happens to the rest of the system when these processes run
- How long the processes run and how they impact the system
- Input data required to start the process, who is providing it, and how they are providing it
- Automatic scheduling of these processes

Typically processes are characterized as:

- **Command-line scripts**—Reflect the data refresh processes and are created based on the time intervals within which they must be executed. Command-line scripts are scheduled by external systems, and prior agreements are made with respect to availability of data. The sample model scripts reflect what typically happens. Because Integrated Operational

Planning is generally in single-user mode, it is important to measure the performance of the scripts and understand how they affect the system. The scripts created by Integrated Operational Planning include: daily, weekly, monthly, and migration scripts.

- **Interactive communication through scenarios**—Analysis types and scenario submissions are often used to communicate among business users. The submission process reflects the end of certain activities.
- **Interactive communication through script templates**—Script templates can be invoked from Integrated Operational Planning and used to model the work flow. Business stakeholders identify the script templates and use the scripts to perform certain activities or to communicate with one another.

Evolution of the Commands Through Releases

- **Version 1.x**—No Model Schema or Edit Schema. Commands to manipulate base schema objects directly and explicit commands for loading data to cubes. No row sources.
- **Version 2.x**—Introduction of row sources and loading through row sources to cubes. Must explicitly load each row source.
- **Version 3.x**—Introduction of the Model Schema and the resulting commands for importing and publishing. Introduction of to simplify script maintenance. Introduction of script templates as first-class objects. Introduction of JACL scripts and the generic Java scripting framework inside the server. (JACL scripts provide flexibility in terms of intermingling `isadmin` commands with access to server resources as well as external access for the Integrated Operational Planning database and other systems.
- **Version 4.x**—Introductions of more , additional scripting language, and more powerful `isadmin` commands.

Scripts

Subtopics

- [JACL Scripts](#)
- [Perl Module Scripts](#)
- [Daily Scripts](#)
- [Weekly Scripts](#)
- [Migration Scripts](#)

JACL Scripts

The following tables list the method details of common bean objects initialized by script execution. The bean and the method are accessed from JACL script as follows:

```
$ISStatementAdmin executeStatement "alter sandbox set batch mode"
```

where `$ISStatementAdmin` is the bean exposed by the framework, `executeStatement` is the method on the bean, and the string is the argument to the method.

Note: See the sample model in *install-root/samples/sample* for examples of different JACL scripts.

Table 7 DBAdmin Method Summary

Bean	Method
Connection	<code>getConnection()</code>
String	<code>getFormattedServerDate()</code>
Date	<code>getServerDate()</code>
long	<code>getServerTime()</code>
String	<code>getServerTimeAsString()</code>
String	<code>toOracleDate(Date date)</code>
Date	<code>toSqlDate(Date date)</code>
String	<code>toSQLServerDate(Date date)</code>
Time	<code>toSqlTime(Date date)</code>
Timestamp	<code>toSqlTimestamp(Date date)</code>

Table 8 EmailAdmin Method Summary

Bean	Method
MailMessage	<code>createMessage(String toUser, String ccUser, String fromUser, String subject, String content, String attachmentPath)</code>

Table 9 ISStatementAdmin Method Summary

Bean	Method
Object	<code>executeStatement(String statement)</code>

Table 10 ModelAdmin Method Summary

Bean	Method
Model	<code>getReadOnly()</code>

Bean	Method
Model	getReadWrite()

Table 11 SandboxAdmin Method Summary

Bean	Method
ListWrapper	getSandboxNamesForResponsesWithChanges(String responseType)

Table 12 SessionAdmin Method Summary

Bean	Method
long	getCurrentSandboxID()
String	getCurrentSandboxName()
boolean	isScriptAccessExclusive(String lockName)
void	setExclusiveScriptAccess(String lockName)

Table 13 TaskAdmin Method Summary

Bean	Method
void	addTask(String taskCmd, String groupName)
boolean	containsPendingTask(String taskCmd, String groupName)
void	deleteTasks(String groupName)
void	executeTasks(String groupName, boolean waitForCompletion)

Perl Module Scripts

You can write wrapper scripts using the Integrated Operational Planning Perl modules to invoke `isadmin` commands.

See `install-root/interlace/perl` for Perl modules available for configuration and runtime support.

See `install-root/samples/sample/bin` for examples of Perl scripts (for example, `dailyscript.pl` and `weeklyscript.pl`) to understand how `isadmin` commands are invoked and how error handling is done.

Daily Scripts

The sample daily script in `install-root/samples/sample` shows how to write Perl and JACL scripts, and how different `isadmin` commands are put together for a certain purpose. This

sample script provides a good framework for you to start your own implementation. The script highlights the following activities:

- Polling for new data
- Importing model objects
- Publishing objects and restructuring
- Loading dimensions
- Loading row sources
- Calculating and submitting the sandbox after loading
- Setting exclusive execution during the script run
- Sending e-mails
- Error handling

Weekly Scripts

The sample weekly script in *install-root/samples/sample* provides a framework to start your implementation and to model the weekly data refresh process flow. The script highlights the following activities:

- All activities that are part of daily script
- Waterfall reporting
- Clearing planning measures from the previous week
- Snapshots
- Archives
- Purging closed and old scenarios
- ETL activity using SQL

Migration Scripts

Before running a migration script, complete the following:

- Run the `preparemigration` command and export all metadata and data from Integrated Operational Planning to the file system.
- Run the `isreset` command to delete the information and create a clean slate for model building.

The migrate script restores the system to its preparemigration state.

The migration script highlights the following:

- The directories where the data resides after `preparemigration`
- Importing the model objects
- Publishing the model objects and restructuring

- Loading the dimension members using the *migration-home/data/rowsource*.csv* files that were exported during preparemigration.
- Using the loader scripts exported as *migration-home/loader/load-dim*.xml*
- Restructuring after a dimension load
- Loading the row source values using the *migration-home/data/rowsource*.csv* files that were exported during preparemigration.
- Using the loader scripts exported as *migration-home/loader/load-rowsource*.xml*
- Loading the cube values using *migration-home/data/cube*.csv* files that were exported during preparemigration. Only user input data from cubes is exported.
- Using the loader scripts exported as *migration-home/loader/load-cube*.xml*
- Creating a sandbox, completing the load, calculating, and submitting
- Restoring the system to its state at the time of preparemigration

See the sample migration script in *install-root/samples/sample* for details.



Writing Formulas

In This Appendix

Keywords.....	171
Literals (Constants)	172
Variables.....	174
Operators	175
OnChange Instructions	176
Formulas for Defining Measures.....	178
Formulas in Constraint Definitions	182
Handling Null (Empty) Cells.....	183

Keywords

Keywords reserved for the language:

- assert
- once
- leading
- true
- false
- and
- or
- not
- null

Note: To specify a string instead of a keyword, put it in quotes.

Literals (Constants)

Subtopics

- [Number Literals](#)
- [Boolean Literals](#)
- [String Literals](#)
- [Dimension Literals](#)
- [Member Literals](#)
- [Dimension Hierarchy Literals](#)
- [Hierarchy Level Literals](#)

This section describes the types of literals.

Note: In case of ambiguity, literals are interpreted based on the context in which they are used. For example, the same symbol may be interpreted as a string, a cell variable, a dimension, or a member.

In the following example, because the function `previous` takes a hierarchy for its parameter, the symbol `Manufacturing` is interpreted as the default hierarchy of the Manufacturing dimension

```
"Ending Inventory Units" [previous(Manufacturing)]
```

Number Literals

Examples of number literals:

- 1
- 100.01
- 0
- null

Note: Null is represented as an empty cell in an Excel spreadsheet.

Boolean Literals

Examples of boolean literals:

- true
- false

String Literals

Examples of string literals:

- “1”
- “string constant”
- null

Dimension Literals

Examples of dimension literals:

- Product
- Part
- Fiscal
- Manufacturing

Member Literals

Examples of member literals:

- Product.KSA

Member KSA in the Product dimension

- Fiscal.”2004 Q1 Week 2”

Member “2004 Q1 Week 2” in the Fiscal dimension

- Fiscal.”FY 2004”.Janurary

Member January for the year 2004 in the Fiscal dimension. because there is more than one member called January (different fiscal years), use name space “FY 2004” to qualify it.

Dimension Hierarchy Literals

Examples of dimension hierarchy literals:

- Fiscal.Fiscal

The hierarchy called Fiscal from dimension Fiscal.

- Fiscal

The default hierarchy of dimension Fiscal, which is Fiscal.Fiscal.

- Fiscal.FiscalQuarter

The hierarchy called FiscalQuarter from dimension Fiscal.

Hierarchy Level Literals

Examples of hierarchy level literals:

- `FiscalQuarter.Quarter` and `Fiscal.Quarter`
because there are two hierarchies, `FiscalQuarter` and `Fiscal`, which both have level `Quarter`, you must qualify the quarter by the hierarchy to which it belongs.
- `Product.Product."Product Family"` and `Product.ProductLine."Product Family"`
because there are two hierarchies, `Product` and `ProductLine`, which both have level `"Product Family"`, you must qualify the quarter by the hierarchy to which it belongs.
- `Product.Model` and `Model`
because there is only one level called `Model`, you can qualify the level with the dimension to which it belongs, or just use the level name.

Variables

Note: A cube to which a measure belongs is called the local cube of the measure, and the measure is called a local measure of the cube

Cell Variables

A cube variable, specified by a measure name, represents a cell location in a cube

If a cell variable defines a measure in a local cube, you need not qualify the measure with the cube name.

In the following example, "Beginning Inventory Units" and "Ending Inventory Units" are local measures; therefore, you need not qualify them by the cube name.

```
"Beginning Inventory Units" = "Ending Inventory  
Units" [previous(Manufacturing)]
```

In the following measure definition, the local cube is `Supply`; therefore, you must qualify the required measure from the `Schedule` cube.

```
"Schedule Required" [level(Week)] = Schedule.required
```

The following cell variable represented by "Dependent Demand Units" represents a cell located at a specific part for a specific time period. Cell variable `source.Required` represents the cell location for the same time period. You must qualify `Required` with `source`; otherwise, it is interpreted as a cell located at the current part and the same time period.

```
"Dependent Demand Units" =  
    "Dependent Demand Units" + source.Required * bomScaleFactor(source.Required)
```

Row Source Variables

A row source variable represents a column from a row source.

In the following example, `ComponentMetrics.standardCost` is a row source variable, where `ComponentMetrics` is a row source name and `standardCost` is a column in that row source.

```
"Scrap Cost" = ComponentMetrics.standardCost * "Scrap Units"
```

To use a row source variable in a formula, you must map the row source to the cube. In the following example, the component column is mapped to a member in the component dimension. `ComponentMetrics` is a time-varying row source. Each record in the row source has an effective time, which is implicitly mapped to the time dimension in the cube. Using this row source mapping, from a cell location, you can find zero or one record from the row source. If the cell location is mapped to one record, the value of `standardCost` is the value of the row source variable. If there is no record to which the cell location is mapped, the variable has a null value.

```
<rm:RowSourceMapping
  keyGeneratorClassName="com.interlacesystems.manufacturing.model.ComponentKeyGenerator">
  <rm:RowSource name="ComponentMetrics"/>
  <rm:TargetDimension name="Component" memberColumn="component"
    namespace="Component">
    <namespace="Component">
  </rm:TargetDimension>
</rm:RowSourceMapping>
```

Operators

Arithmetic and Boolean Operators

Table 14 Operators and Their Meanings

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
=	Equal to
<>	Not equal to
not	Logical negation

Operator	Meaning
and	Logical and
or	Logical or

Location Operators

Location operators follow cell variables and are represented as brackets.

Location operators on the left side of a formula define the cell locations at which the formula is defined or applied. You can define only one formula per cell location.

In the following example, the location operator in "Schedule Required" [level(Week)] defines the formula for cell locations at a weekly level.

```
"Beginning Inventory Units" = "Ending Inventory Units"[previous(Manufacturing)]
```

```
"Schedule Required"[level(Week)] = Schedule.required
```

```
"Demand Units Avg"[level(Week)] = avg("Demand Units"[sibling (Manufacturing)])
```

Location operators on the right side of a formula turn a single cell location into a collection of locations.

In the previous example, "Ending Inventory Units" [previous(Manufacturing)] turns into the cell location for the same part but at the previous time period. "Demand Units" [sibling (Manufacturing)] turns into the cell locations that directly reside under the same time period. In this case, it is the collection of cell locations of "Demand Units" for the same part and at all the weeks of the current month.

OnChange Instructions

OnChange instructions are used in formulas for roll-up or roll-down operations, where changes from blocks at one sparse level are propagated to blocks at another sparse level.

The following roll-down formula computes a percentage of the Forecast Units in the parent member and adds that value to the Forecast Units in the child member. This happens along a sparse dimension. In other words, the value is allocated to different children using a percentage.

```
"Forecast Units" = "Forecast Units" +  
pctof(ProductAllocation.percentage, deltaValue(source("Forecast  
Units")))
```

OnChange instructions include:

- [deltaValue](#)
- [dependencyAttribute](#)
- [isInSourceScope](#)
- [newValue](#)
- [oldValue](#)

- [source](#)
- [sourceMember](#)
- [sourceSparseMember](#)

deltaValue

`deltaValue` returns the delta value of a changed entry at the specified location. The changes are stored in change entries as old, new and deltas. `deltaValue` is applicable only to change entries or “onChange” formulas.

deltaValue(Location location)

dependencyAttribute

`dependencyAttribute` returns the value of the attribute in the dependency between `sourceMember` and the corresponding target member in the block currently processed along the dimension. The `sourceMember` in the dimension for the current operation is determined from a block or a change entry.

dependencyAttribute(Dimension d, String attributeName)

isInSourceScope

`isInSourceScope` checks whether the specified location in a roll-up or roll-down operation is in the slice represented by the operation's source object. For roll-up, the source is inferred as the child, and for roll-down the source is inferred as the parent. `isInSourceScope` is applicable only in roll-up or roll-down operations where changes are propagated from a source object.

isInSourceScope(Location location)

newValue

`newValue` returns the new value of a changed entry at the specified location. The changes are stored in change entries as old, new and deltas. `newValue` is applicable only to change entries or “onChange” formulas.

newValue(Location location)

oldValue

`oldValue` returns the old value of a changed entry at the specified location. The changes are stored in change entries as old, new and deltas. `oldValue` is applicable only to changed entries or “onChange” formulas.

oldValue(Location location)

source

`source` returns the source of the specified location. The location is cast into a source location, and the value for that location is fetched from the appropriate place. For example, the value of “location” is the value from the current block data cell, whereas the value of “source(location)” is the value from the location with the member replaced by its corresponding source member along the dimension on which roll-up or roll-down is performed. `source` is applicable only in roll-up or roll-down operations where changes are propagated from a source object.

source(Location loc)

sourceMember

sourceMember takes the current location in the current context and the source object (block or a change entry) and determines the member in source object along the given dimension. **sourceMember** is applicable only in roll-up or roll-down operations where changes are propagated from a source object.

sourceMember(Dimension d)

sourceSparseMember

sourceSparseMember takes the current location in the current context and the source object (block or a change entry) and determines the member in the source object along the given sparse dimension. **sourceSparseMember** is applicable only in roll-up or roll-down operations where changes are propagated from a source object.

sourceSparseMember(Dimension d)

Formulas for Defining Measures

Formulas are used in measure and constraint definitions. They consist of row source variables, cube locations, and functions. Formulas are restricted by the dense and sparse dimensions of the cube to which they belong.

The computation of a measure consists of the following elements:

- Dense calculation
- Aggregation along dense dimensions
- Sparse calculation along sparse dimensions or between cubes

Dense calculations are defined by formulas, and sparse calculations are defined by batch formulas and on-change formulas. Sparse calculations can be roll-up or allocation.

Every formula has a scope. For example, a dense formula and its scope define a family of formulas for the cubular cells in the scope. The dense formula is a multidimensional formula, and the formulas in the family are expanded formulas.

When writing formulas, keep in mind the following points:

- **Formula scope can be defined explicitly or inferred from other calculation elements.**

In the following example, *Beginning Inventory Units* is a loaded measure. It has a dense formula and dense aggregation along the Manufacturing dimension. *Beginning Inventory Units* belongs to a cube that has two dimensions, Component (sparse) and Manufacturing (time/dense). As a result, the scope of the dense formula applies to the following members:

- All members in the Component dimension (there is no sparse calculation)
- Members at the weekly level in the Manufacturing dimension (there is a dense roll-up from week along that dimension)

- Each cell of the Beginning Inventory Units measure where no loaded value exists (loaded values override calculated values)

```
<measure:Measure name="Beginning Inventory Units"
  scale="0" type="loaded" styleName="unitStyle">
  <measure:Formula>
    <![CDATA["Beginning Inventory Units"="Ending Inventory
      Units" [previous (Manufacturing)]]]>
  </measure:Formula>
  <measure:DataRowSource name="Inventory" columnName="quantity"/>
  <measure:SimpleRollUp summaryOperator="first_in_period">
    <measure:Dimension name="Manufacturing"/>
  </measure:SimpleRollUp>
</measure:Measure>
```

Note: If a cell is modified by a user input value, the formula is still applicable to the cell. Integrated Operational Planning generates a calculation exception that denotes the discrepancy between the calculated value based on the formula and the user input value. A loaded value is considered a calculated value. If there is a difference between a loaded value and a calculated value from the formula, it is not considered as a discrepancy.

In the following example, Ending Inventory Units is a derived measure. It belongs to a cube that has two dimensions, Component (sparse) and Manufacturing (time/dense). Its scope applies to the following members:

- All members in the Component dimension (there is no sparse calculation)
- Members at the weekly level in the Manufacturing dimension (there is a dense roll-up from week along that dimension)

```
<measure:Measure name="Ending Inventory Units"
  scale="0" type="derived" default="true" styleName="unitStyle">
  <measure:Formula>
    <![CDATA["Ending Inventory Units"="Beginning Inventory
      Units" + Receipts - "Total Demand Units"]]>
  </measure:Formula>
  <measure:SimpleRollUp summaryOperator="last_in_period">
    <measure:Dimension name="Manufacturing"/>
  </measure:SimpleRollUp>
</measure:Measure>
```

- The scope of a cross-cube formula is defined explicitly by the cube map.

In the following example, the cross-cube formula scope applies to the following members:

- All members in the Component dimension that fall in the range of the cube mapping functions
- Members at the weekly level in the Manufacturing dimension (there is a dense roll-up from week along that dimension)

Note: In this example, the on-change formula has the same scope as the cross-cube formula.

```

<measure:Measure name="Schedule Required" hidden="true"
  scale="0" type="derived">
  <measure:SimpleRollUp summaryOperator="sum">
    <measure:Dimension name="Manufacturing">
</measure:SimpleRollUp>
<measure:ComplexRollUp>
  <measure:Cube name="Schedule"/>
  <measure:Formula>
    <![CDATA["Schedule Required"=Schedule.required]]>
  </measure:Formula>
  <measure:OnChangeFormula>
    <measure:Measure name="required"/>
    <measure:Formula>
      <![CDATA["Schedule Required"="Schedule Required"
        + deltaValue(Schedule.required)]]>
    </measure:Formula>
  </measure:OnChangeFormula>
</measure:ComplexRollUp>
</measure:measure>

```

- The scope of a roll-up formula along a sparse dimension excludes leaf members.

In the following example, the roll-up formula and the on-change formula for the roll-up are only applicable to Component members at a non-leaf level.

```

<measure:Measure name="Dependent Demand Units"
  scale="0" type="derived" styleName="unitStyle">
  <measure:ComplexRollUp>
    <measure:Dimension name="Component"/>
    <measure:Formula>
      <![CDATA{"Dependent Demand Units"="Dependent Demand
        Units" + source(Required) *
        bomScaleFactor(sourceSparseMember(Component))]]>
    </measure:Formula>
    <measure:OnChangeFormula>
      <measure:Measure name="Required"/>
      <measure:Formula>
        <![CDATA["Dependent Demand Units"=Dependent
          Demand Units" + deltaValue(source(Required)) *
          bomScaleFactor(sourceSparseMember(Component))]]>
      </measure:Formula>
    </measure:OnChangeFormula>
  </measure:ComplexRollUp>
  <measure:SimpleRollUp summaryOperator="sum">
    <measure:Dimension name="Manufacturing"/>
  </measure:SimpleRollUp>
</measure:Measure>

```

- Use startLevel to restrict the roll-up scope.

In the following example, a measure is defined in the Forecast cube. The roll-up formulas are applicable to members in the Geography dimension at the Region level and to the upper levels.

```

<measure:Measure name="Marketing Forecast Revenue"
  displayName="Marketing Forecast $"
  scale="2" type="loaded" styleName="dollarStyle">
  <measure:DataRowSource name="MktForecast" columnName="REVENUE"/>
  <measure:SimpleRollUp summaryOperator="sum">

```

```

        <measure:Dimension name="ProductLine"/>
    </measure:SimpleRollUp>
    <measure:SimpleRollUp summaryOperator="first_in_period">
        <measure:Dimension name="Fiscal"/>
    </measure:SimpleRollUp>
    <measure:ComplexRollUp startLevel="Region">
        <measure:Dimension name="Geography"/>
        <measure:Formula>
            <![CDATA[{"Marketing Forecast Revenue"="Marketing
Forecast Revenue" + source("Marketing Forecast
Revenue")}]>
        </measure:Formula>
        <measure:OnChangeFormula>
            <measure:Measure name="Marketing Forecast Revenue"/>
            <measure:Formula>
                <![CDATA["Marketing Forecast Revenue"="Marketing
Forecast Revenue" + deltaValue
(source("Marketing Forecast Revenue"))]]>
            </measure:Formula>
        </measure:OnChangeFormula>
    </measure:ComplexRollUp>
</measure:Measure>

```

- **The inference of a roll-down scope is the inverse of a roll-up scope.**

```

<measure:Measure name="Marketing Forecast Revenue"
    displayName="Marketing Forecast $"
    scale="2" type="loaded" styleName="dollarStyle">
    <measure:DataRowSource name="MktForecast" columnName="REVENUE"/>
    <measure:SimpleRollUp summaryOperator="sum">
        <measure:Dimension name="ProductLine"/>
    </measure:SimpleRollUp>
    <measure:SimpleRollUp summaryOperator="first_in_period">
        <measure:Dimension name="Fiscal"/>
    </measure:SimpleRollUp>
    <measure:ComplexRollUp startLevel="Region">
        <measure:Dimension name="Geography"/>
        <measure:Formula>
            <![CDATA["Marketing Forecast Revenue"="Marketing
Forecast Revenue" + sourc
("Marketing Forecast Revenue")]]>
        </measure:Formula>
        <measure:OnChangeFormula>
            <measure:Measure name="Marketing Forecast Revenue"/>
            <measure:Formula>
                <![CDATA["Marketing Forecast Revenue"="Marketing
Forecast Revenue" + deltaValue
(source("Marketing Forecast Revenue"))]]>
            </measure:Formula>
        </measure:OnChangeFormula>
    </measure:ComplexRollUp>
</measure:Measure>

```

- **Scope operators on sparse dimensions are only on the left hand side of the measure formula.**

In the following example, the Loaded Projected ASP measure is restricted by the [level (Month)] operator, and the Backlog Units measure is restricted by the

[previous(Fiscal)] operator. The two operators are on the Time dimension, which is dense.

```
<measure:Measure name="Loaded Projected ASP"
  displayName="Projected ASP"
  scale="2" type="derived">
  <measure:Formula>
    <![CDATA["Loaded Projected ASP"[level(Month)]=ProjectedASP.asp]]>
  </measure:Formula>
</measure:Measure>

<measure:Measure name="Backlog Units" scale="0" type="derived">
  <measure:Description>
    Booked units minus shipped units in a given time period
  </measure:Description>
  <measure:Formula>
    <![CDATA[["Backlog Units"="Backlog Units"[previous(Fiscal)]+"Booked
Units"- "Shipped Units"]]>
  </measure:Formula>
  <measure:SimpleRollUp summaryOperator="last_in_period">
    <measure:Dimension name="Fiscal"/>
  </measure:SimpleRollUp>
  <measure:SimpleRollUp summaryOperator="sum">
    <measure:Dimension name="Product"/>
  </measure:SimpleRollUp>
  <measure:Dimension name="Product"/>
</measure:Measure>
```

Tip: To avoid ambiguity when using names that could imply multiple objects (dimensions, members, hierarchies, or levels) in a measure formula, use a casting function to convert to the correct type.

In the following example, “dimension(Mode)” converts the name “Mode” to a dimension object.

```
"Projected Fuel Surcharge" =
if (memberName(dimension(Mode)) == "TRUCK",

"Projected Fuel Surcharge Rate",
"Projected Line Haul Rate"
)
```

Other casting functions include `cube`, `dimensionMember`, `hierarchy`, `hierarchyLevel`, and `location`. See [“Member Functions” on page 202](#).

Formulas in Constraint Definitions

Constraint formulas inside `<definition>` tags define where and when exceptions occur. Use “assert once” to report exceptions at the beginning of each consecutive time period, “assert” to report exceptions on every occurrence, and “assert leading” to report exceptions on the first occurrence.

The following is an example of a `Part_Shortage` constraint. The constraint formula inside the `<definition>` tags defines the exception in the context of “Ending Inventory Units” at a weekly

level. The formula defines to not report an exception if the current part is a finished good, or if the current location is in the past, or if the current quarter is in the future or the past, or if there is a positive quantity in “Ending Inventory Units.” The phrase “assert once” defines to report exceptions at the beginning of each consecutive time period.

```
<Constraint name="Parts_Shortage"
  cibe="Supply"
  description="Parts shortage exists based on current demand"
  measure="Ending inventory Units"
  owner="mfg_ops"
  dueDate="P2D"
  priority="High"
  type="Shortage">
  <Definition><![CDATA[
    Parts_Shortage."Ending Inventory Units"[level(Week)]
    assert once(ComponentMetrics.isFinishedGood or
      is Past() or
      not isCurrent(parent(parent("Ending Inventory Units", Manufacturing),
Manufacturing)) or
      "Ending Inventory Units"<=0)
  ]]>/Definition>
  <Context>
    <Value name="Part" type="string" length="100">ComponentMetrics.part="Part"</Value>
    <Value name="Location" type="string" length="100">ComponentMetrics.location</Value>
    <Value name="EndingInventoryUnits" displayName="Ending Inventory Units"
type="number">
      "Ending Inventory Units"</Value>
    <Value name="TotalDemandUnits" displayName="Total Demand Units" type="number">
      "Total Demand Units"</Value>
  </Context>
</Constraint>
```

The expressions inside the Context tags specify what data need to be captured when an exception occurs.

Handling Null (Empty) Cells

In Integrated Operational Planning, empty cells are represented by `java.lang.Double.NaN`. In many cases, the value behaves like zero, very differently from how NaN behaves based on a Java language specification.

Table 15 Differences in Computation Environments

	Java	SQL	Excel	Integrated Operational Planning
<code>1 > null</code>	false	false	true	true
<code>1 >= null</code>	false	false	true	true
<code>1 < null</code>	false	false	false	false
<code>1 <= null</code>	false	false	false	false

	Java	SQL	Excel	Integrated Operational Planning
1 == null	false	false	false	false
1 != null	true	false	true	true
null == null	false	false	true	true
null != null	true	false	false	false
null + null	null	null	0	null
null - null	null	null	0	null
null * null	null	null	0	null
null / null	null	null	#DIV/0!	null
1 / 0	Exception	Exception	#DIV/0!	null
1 / null	null	null	#DIV/0!	null
1 + null	null	null	1	1
1 - null	null	null	1	1
1 * null	null	null	0	null
0 == null	false	false	true	true
0 != null	true	false	false	false
-1 > null	false	false	false	false
-1 >= null	false	false	false	false
-1 < null	false	false	true	true
-1 <= null	false	false	true	true
-1 == null	false	false	false	false
-1 != null	true	false	false	false
-1 + null	null	null	-1	-1
-1 - null	null	null	-1	-1
-1 * null	null	null	0	null
sum(1, null)		1	1	1
sum(-1, null)		-1	-1	-1
avg(1, null, -1, 6)		2	2	2
1 > null w/ORDER BY ASC		true	true	

	Java	SQL	Excel	Integrated Operational Planning
-1 > null w/ORDER BY ASC		false	false	
0 < null w/ORDER BY DESC		true	false	
-1 < null w/PRDER BY DESC		true	false	
1 < null w/ORDER BY DESC		false	false	
-1 < null w/ORDER BY DESC		true	true	
0 < null w/ORDERBY DESC		true	false	
null = = w/GROUP BY		true	true	



Functions

In This Appendix

Basic Functions	188
Mapping Functions.....	194
Math Functions.....	199
Member Functions	202
Shape Functions	220
Time Functions	227
External Functions	232
Custom Functions.....	233

See also [Appendix D, “MDX Extensions”](#).

Basic Functions

Subtopics

- `abs`
- `add`
- `and`
- `avg`
- `ceiling`
- `div`
- `doubleValue`
- `eq`
- `floor`
- `ge`
- `getPositive`
- `gt`
- `isBlank`
- `isNull`
- `isNullString`
- `isZero`
- `le`
- `lt`
- `max`
- `min`
- `mul`
- `ne`
- `not`
- `nval`
- `or`
- `pct`
- `pctof`
- `random`
- `round`
- `streq`
- `string`
- `stringValue`
- `sub`
- `sum`
- `toBoolean`
- `toDouble`

Basic functions include arithmetic functions, boolean functions, relational functions, and basic string functions. These functions are primarily used in the construction of formulas for derived measures.

abs

abs returns the absolute value of a.

```
abs(double a)
```

add

add returns a+b.

```
add(double a, double b)
```

and

and returns a and b.

```
and(boolean a, boolean b)
```

avg

Subtopics

- `avg(double[] addends)`
- `avg(double[] addends, boolean includeEmptyCells)`

avg(double[] addends)

avg returns the average of an array of double numbers.

```
avg(double[] addends)
```

avg(double[] addends, boolean includeEmptyCells)

avg returns the average of the numbers, given an array of double numbers. The boolean flag **includeEmptyCells** specifies whether the Double.NaN in the array should be considered for average.

```
avg(double[] addends, boolean includeEmptyCells)
```

ceiling

ceiling, if the number is null, returns null; otherwise, rounds the double to the nearest integer.

```
ceiling(double a)
```

div

div returns a/b.

div(double a, double b)

doubleValue

doubleValue forces the evaluation of a measure to a double. Used when a measure value needs to be passed to a function that takes *object* as an argument type.

doubleValue(double d)

eq

eq returns true if a=b.

eq(double a, double b)

eq(date a, date b)

floor

floor, if the number is null, returns null; otherwise, floors the double to the nearest integer.

floor(double a)

ge

ge returns true if a≥b.

ge(double a, double b)

getPositive

getPositive, given an array of doubles, returns a positive number. The boolean flag “firstToLast” controls whether to search the array for first to last or from last to first.

getPositive(double[] operands, boolean firstToLast)

gt

gt returns true if a>b.

gt(double a, double b)

isBlank

isBlank returns true if null. Used for loaded or user input cells, not for cells calculated with measure or aggregation formulas.

isblank(double a)

isNull

isNull returns true if null.

isNull(double a)

isNullString

isNullString returns true if null.

isNullString(string a)

isZero

isZero returns true if null or zero.

isZero(double a)

le

le returns true if $a \leq b$.

le(double a, double b)

lt

lt returns true if $a < b$

lt(double a, double b)

max

Subtopics

- `max(double a, double b)`
- `max(double[] mbrs)`

max(double a, double b)

max returns the larger of {a, b}.

max(double a, double b)

max(double[] mbrs)

max returns the largest of the set.

max(double[] mbrs)

min

Subtopics

- `min(a, b)`
- `min(double[] mbrs)`

min(a, b)

`min` returns the smaller of (a, b).

min(double a, double b)

min(double[] mbrs)

`min` returns the smallest of the set.

min(double[] mbrs)

mul

`mul` returns `a*b`, or null if either is null.

mul(double a, double b)

ne

`ne` returns true if a does not equal b.

ne(double a, double b)

not

`not` returns values that are not a.

not(boolean a)

nval

`nval` returns `ifnull` if the value is null; otherwise, returns the value.

nval(double value, double ifnull)

or

`or` returns a or b.

or(boolean a, boolean b)

pct

pct returns the fraction as a percentage. For example, $(a/b)*100$.

pct(double a, double b)

pctof

pctof returns a as a percentage of b. For example, $(a/100)*b$.

pctof(double a, double b)

random

random returns a random double number.

random()

round

round—If the number is null, returns null; otherwise, rounds the double to the nearest integer.

round(double a)

streq

streq returns true if string a is equal to string b.

streq(string a, string b)

string

string converts an argument to string. The argument could be a measure, cube, dimension, hierarchy, or level.

string(string a)

stringValue

stringValue, if the flag is true, returns a string in a double format, for example, 1.0; otherwise returns an integer format, for example, 1.

stringValue(double d, boolean flag)

sub

sub returns $a - b$.

sub(double a, double b)

sum

sum returns the sum of values passed as an argument.

```
sum(double[] addends)
```

toBoolean

toBoolean returns true if the specified string is **true** or **on** or **yes** (cases are ignored).

```
toBoolean(string s)
```

toDouble

toDouble converts the string to a double. If formatting problems exists, or if the specified string is not a number, then Double.NaN is returned. Returns a double. Converts a string passed as an argument to a double.

```
toDouble(string s)
```

Mapping Functions

Subtopics

- [Time Dimension Mapping Functions](#)
- [Dimension Member Attribute Name Mapping Functions](#)
- [Member Name Mapping Functions](#)
- [Row Source Mapping Functions](#)
- [Splicing and Splitting Mapping Functions](#)

Mapping functions are used in cube maps, which map dimensions, levels, and members between cubes.

Time Dimension Mapping Functions

Subtopics

- [calendarMap](#)
- [overlapCalendarMap](#)
- [timeMemberMap](#)

calendarMap

calendarMap, given a time member and a target dimension, returns the closest mapped time member in the target dimension, which is also a (different) time-dimension. Time ranges associated with the time members are used to find the closet map.

```
calendarMap<ArrayList<Member> current, ArrayList<String> extraParams, Dimension d,  
String namespace)
```

overlapCalendarMap

`overlapCalendarMap` returns the closest mapped time member in the target dimension. Similar to `calendarMap` except you can have multiple months or other periods overlapping a target member.

```
overlapCalendarMap(ArrayList<Member> current, ArrayList<String> extraParams, Dimension  
d, String namespace)
```

timeMemberMap

`timeMemberMap(dimension)`

`timeMemberMap(dimension)`, given a dimension, returns the closest matched member of the current time member in the dimension. The closest-match maximizes the criteria (intersection/memberPeriod). For example, if a quarter and a year intersect the source member, quarter has a better value for the criteria and is therefore returned.

```
timeMemberMap(Dimension d)
```

`timeMemberMap(dimension, member)`

`timeMemberMap(dimension, member)`, given a time member, and a dimension, returns the closest matched member of the given member in the dimension. The closest-match maximizes the criteria (intersection/memberPeriod). For example, if a quarter and a year intersect the source member, quarter has a better value for the criteria and is therefore returned.

```
timeMemberMap(Member member, Dimension d)
```

Dimension Member Attribute Name Mapping Functions

Subtopics

- [attributeMap](#)
- [attributeReverseMap](#)

attributeMap

`attributeMap`, given information about the source dimension member and attribute name, returns the target dimension member. The attribute string value is expected to match the target dimension member names. The namespace and the dimension name are used to construct the fully qualified target dimension member name.

This function returns an arraylist containing members in the result set. If there is no mapping, an empty list is returned.

```
attributeMap(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)
```

where `current` is the source dimension member name that is current being mapped, `d` is the target dimension, and `namespace` is the namespace in the target dimension used to construct the result member list.

`attributeMap` is defined in the cube map as follows:

```
attributeMap(source_dimension_name, attribute_name)
```

In the following example, the source dimension member is passed in the `current` list. The `extraParams` string list contains “Associated Geography” (attribute name).

```
attributeMap(Plants, Associated Geography)
```

If a string contains a space, do not quote it. In the previous example, Associate Geography is not quoted.

attributeReverseMap

`attributeReverseMap`, given information about target dimension member and attribute name for the source dimension members, returns the source dimension member. The target member name is expected to match the attribute value of one or more source dimension members. A reverse map is performed on the attribute value to find the name of the member in the source dimension with that value for their attribute. The namespace and dimension name are used to construct the fully qualified source dimension member name. The level name is used to narrow the reverse map search.

This function returns an arraylist containing members in the result set. If there is no mapping, an empty list is returned.

```
attributeReverseMap(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)
```

where `current` is the source dimension member name that is current being mapped, `d` is the target dimension, and `namespace` is the namespace in the target dimension used to construct the result member list.

How `attributeMap` is defined in the cube map:

```
attributeMap(source_dimension_name, attribute_name)
```

In the following example, the source dimension member is passed in the `current` list. The `extraParams` string list contains “Associated Geography”(attribute name).

```
attributeMap(Plants, Associated Geography)
```

If a string contains a space, do not enclose it in quotation marks.. In the previous example, Associate Geography is not enclosed in quotation marks.

Member Name Mapping Functions

Subtopics

- [exactMap](#)
- [oneToOneMap](#)

exactMap

`exactMap`, given a set of members, constructs a new array list and returns the member as the result list

```
exactMap(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)
```

oneToOneMap

`oneToOneMap`, given a set of members, takes *only* the name, namespace, the target dimension and constructs a fully qualified member name in the target dimension. If the dimension contains the member, then returns that as a result; otherwise returns an empty array.

```
oneToOneMap(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)
```

Row Source Mapping Functions

Subtopics

- [rowsourceLookup](#)
- [rowsourceQuery](#)

rowsourceLookup

`rowsourceLookup`, given information about source dimension member, row source name, and lookup columns, returns the target dimension member. Uses key lookup to return an array list containing one member in the result set. If there is no mapping, an empty list is returned.

```
rowsourceLookup(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)
```

where `current` is the source dimension member name that is currently being mapped, `d` is the target dimension, and `namespace` is the namespace in the target dimension used to construct the result member list.

`rowsourceLookup` is defined in the cube map as follows:

```
rowsourceLookup(source_dimension_name, rowsource_name, lookup_column)
```

In the following example, the source dimension member is passed in the `current` list. The `extraParams` string list contains “geographyDC”(row source name) and “dcName”(lookup

column name) in that order. The key column of the row source “geographyDC” is the “source member name.”

```
rowsourceLookup(Geography, geographyToDC, dcName)
```

extraParams can contain more than two strings passed to the row source key. The first string in extraParams is the row source name and the last string is the lookup column for the target member name, the source member name, and the intermediate strings in extraParams for the row source key.

If a string contains a space, do not enclose it in quotation marks. In the following example, the lookup column “Dist Center Name” is not enclosed in quotation marks.

```
rowsourceLookup(Geography, geographyToDC, Dist Center Name)
```

If a row source is time-varying, the first element is used as the lookup value.

rowsourceQuery

rowsourceQuery, given information about the source dimension member, row source name, and lookup columns, returns the target dimension members. Uses an index to return an array list containing one or more members in the result set. If there is no mapping, an empty list is returned.

```
rowsourceQuery(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)
```

where current is the source dimension member name that is currently being mapped, d is the target dimension, and namespace is the namespace in the target dimension used to construct the result member list.

rowsourceQuery is defined in the cube map as follows:

```
rowsourceQuery(source_dimension_name, rowsource_name, index_name, lookup_column)
```

In the following example, the source dimension member is passed in the current list. The extraParams string list contains “geographyDC”(row source name), “gcIndex1”(index name), and “dcName”(lookup column name) in that order. The index columns of the row source “geographyDC” is the “source member name.”

```
rowsourceQuery(Geography, geographyToDC, gcDCIndex1, dcName)
```

extraParams can contain more than two strings passed to the row source key. The first string in extraParams is the row source name and the last string is the lookup column for the target member name, the source member name, and the intermediate strings in extraParams for the row source index.

If a string contains a space, do not enclose it in quotation marks. In the following example, the lookup column “Dist Center Name” is not enclosed in quotation marks.

```
rowsourceQuery(Geography, geographyToDC, gcDCIndex1, Dist Center Name)
```

If a row source is time-varying, the first element is used as the lookup value.

Splicing and Splitting Mapping Functions

Subtopics

- `splice`
- `split`

splice

`splice`, given a set of members, splices them together. The argument `extra Params` is not used. the members passed in the argument `current` are spliced together.

splice(`ArrayList<Member> current`, `ArrayList<String> extraParams`, `Dimension d`, `String namespace`)

split

`split`, given a set of members, splits them into separate members and return the member that exists in the dimension passed as an argument.

split(`ArrayList<Member> current`, `ArrayList<String> extraParams`, `Dimension d`, `String namespace`)

Math Functions

Subtopics

- `cos`
- `exp`
- `factorial(int x)`
- `gcd`
- `inflatedValue`
- `log`
- `log10`
- `mod`
- `power`
- `sin`
- `sqrt`
- `tan`

Math functions include trigonometric functions, exponential functions, and logarithmic functions, primarily used in formulas for derived measures.

COS

`cos` returns the cosine of a number. Value considered in radians.

cos(`double a`)

exp

`exp` returns e raised to the power of a given number.

```
exp(double a)
```

factorial(int x)

`factorial` returns the factorial of a nonnegative integer n , denoted by $n!$, the product of all positive integers less than or equal to n . For example, $5! = 5 \times 4 \times 3 \times 2 \times 1$

```
factorial(int x)
```

gcd

`gcd` returns the greatest common divisor, which is the largest common factor between two numbers.

```
gcd(int n, int d)
```

inflatedValue

`inflatedValue` computes the inflation adjusted value given a base value, inflation rate, and the number of time periods. The formula is $b \cdot (1+i)^k - k \cdot (1-i)^k / (1-i)$ where b is the base-value, i is the inflation-rate, and k is the number of time-periods from current period.

```
inflatedValue(double baseValue, double inflationRate, double timePeriods)
```

log

`log` returns the logarithm of the given number to the base 2.

```
log(double a)
```

log10

`mod` returns the modulus(remainder) of a/b .

```
mod(double a, double b)
```

`log10` returns the logarithm of the given number to the base 10.

```
log10(double a)
```

mod

`mod` returns the modulus(remainder) of a/b .

```
mod(double a, double b)
```


power

`power` returns the result of a number raised to a power.

power(double a, double b)

sin

`sin` returns the sine of a number..Value considered in radians.

sin(double a)

sqrt

`sqrt` returns the square root of the given number.

sqrt(double a)

tan

`tan` returns the tangent of a number. The input is assumed to be in radians.

tan(double a)

Member Functions

Subtopics

- [Defining Member Functions](#)
- [childMembers\(location hierarchy\)](#)
- [children](#)
- [count](#)
- [cousinLocations](#)
- [cube](#)
- [currentValue](#)
- [dateMemberProperty](#)
- [dateProperty](#)
- [descendants](#)
- [dimension](#)
- [dimensionMember](#)
- [doubleProperty](#)
- [equals](#)
- [exists](#)
- [firstChild](#)
- [firstDescendant](#)
- [hierarchy](#)
- [hierarchyLevel](#)
- [in](#)
- [indexOf](#)
- [is Functions](#)
- [lastChild](#)
- [lastDescendant](#)
- [location](#)
- [lookup functions](#)
- [member](#)
- [memberName](#)
- [nextMember](#)
- [memberProperty](#)
- [parent](#)
- [parentMembers](#)
- [parents](#)
- [pastCousinLocations](#)
- [previousMember](#)
- [property](#)
- [range](#)
- [siblingCount](#)
- [siblingLocations](#)

Defining Member Functions

Member functions query for properties of the multidimensional cube structure and other objects

upon which the underlying system is built. The functions query for child members, parent members, member names, and members levels. These functions are used in formulas for computing derived measures, and they define the scope of computations and mappings. The functions are either evaluated statically at compile time (and appropriate shortcuts are made), or they are evaluated at runtime.

In the following formula, when the system compiles and generates code for the measure in its cube, it knows the structure of the cube, the cube levels, and the cube dimensions:

```
measure1isLeaf = if(isLeaf(), measure2, measure3)
```

For those members that are leaf members, the following code is generated:

```
measure1 = measure2
```

For those members that are not leaf members, the following code is generated:

```
measure1 = measure3
```

The `isLeaf` function in this example is compiled statically.

See [“Creating Cube-to-Cube Maps” on page 99](#)

childMembers(location hierarchy)

`childMembers`, given a member and a dimension, returns all child members along with the dimension. (Children are defined in terms of calculation dependencies.)

```
childMembers(ArrayList<Member> current, ArrayList<String> extraParams, Dimension d,  
String namespace)
```

children

Subtopics

- [children\(hierarchy\)](#)
- [children\(location hierarchy\)](#)
- [children\(member, hierarchy\)](#)

children(hierarchy)

`children(hierarchy)` returns the array of child members, who are the children of the current location along the hierarchy passed as the argument.

```
children(Hierarchy h)
```

children(location hierarchy)

`children(location hierarchy)` returns the array of child members, who are the children of the location along the hierarchy passed as the argument.

```
children(Location loc, Hierarchy h)
```

children(member, hierarchy)

`children(member, hierarchy)` returns the array of child members, who are the children of the location along the hierarchy passed as the argument.

children(Member m, Hierarchy h)

count

`count` returns the number of elements in a supplied list. If the list is null, returns NA.

count(Location[] list)

cousinLocations

Subtopics

- `cousinLocations(past time members)`
- `cousinLocations(root member)`

cousinLocations(past time members)

`cousinLocations(past time members)`, given an ancestor member, a location, and a hierarchy for a dimension, the cousin location is where the member in the dimension of the specified location is replaced with members that are descendants of the **ancestor** member but at the same level as the member in the specified location. It is assumed that the dimension of the ancestor member is same as that of the specified hierarchy. The locations are ordered from left to right along the level of the specified location. The boolean flags control which locations to ignore (to the left or to the right). The `ensureIsPast` boolean flag controls whether to return only the locations with past time members.

cousinLocations(Location loc, Member ancestor, Hierarchy h, boolean excludeLeftCousins, boolean excludeRightCousins, boolean excludeSelf, boolean ensureIsPast)

cousinLocations(root member)

`cousinLocations(root member)`, given a **root** member (implicit ancestor), a location, and a hierarchy for a dimension, the cousin location is where the member in the dimension of the specified location is replaced with members that are descendants of the **ancestor** member but at the same level as the member in the specified location. It is assumed that the dimension of the ancestor member is same as that of the specified hierarchy. The locations are ordered from left to right along the level of the specified location. The boolean flags control which locations to ignore (to the left or to the right).

cousinLocations(Location loc, Hierarchy h, boolean excludeLeftCousins, boolean excludeRightCousins, boolean excludeSelf)

cube

`cube` casts a cube into a cube. Used to resolve ambiguity resulting from two or more function signatures with the same name. For example, assume that you have two function signatures. One takes a cube and the other takes a measure. If the name of the cube and the measure are the same, then invoking that function in a formula creates ambiguity. By explicitly casting the string with the `cube` function, the ambiguity is resolved.

cube (cubeObj)

currentValue

`currentValue` returns the value of the current location.

You can use `currentValue` to say “do not change the value.” `currentValue` has a dynamic result to avoid the infinite loop that may occur if you wrote the formula directly. For example, `x=if(condition, y,x)` (if the condition is true, then set x to y; otherwise, leave x alone) may cause an infinite loop during dependency checking:

currentValue ()

dateMemberProperty

`dateMemberProperty` returns the attribute value (expected to be Date type) with the specified name from the specified member. Returns null if the member is null or if the attribute is not found.

dateMemberProperty (Member m, String attributeName)

dateProperty

`dateProperty` returns the value of the attribute (expected to be of Date type) with the specified name from the member in the current context along the specified dimension. Returns null if the member is null or if the attribute is not found.

dateProperty (Dimension d, String attributeName)

descendants

`descendants`, given a member and a dimension, returns all child members along the dimension. (Children are defined in terms of calculation dependencies.)

descendants (ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)

dimension

`dimension` casts a name into a dimension

dimension(String name)

dimensionMember

dimensionMember casts a member into a member. Used to resolve ambiguity resulting from function signatures with the same name. For example, assume you have two function signatures. One takes a hierarchy and the other takes a member. If the name of the hierarchy and the member are the same, then invoking that function in a formula creates ambiguity. By explicitly casting the string with the **dimensionMember** function, the ambiguity is resolved.

dimensionMember(member)

doubleProperty

doubleProperty returns the double attribute of the member with the specified property name.

doubleProperty(Member m, String propertyName)

equals

equals returns true if m1=m2

equals(Member m1, Member m2)

exists

exists checks whether the specified location is valid. (An invalid location, for example, contains invalid members or time members that fall outside the specified horizon.)

exists(Location location)

firstChild

Subtopics

- [firstChild\(hierarchy, location\)](#)
- [firstChild\(hierarchy, member\)](#)
- [firstChild\(hierarchy\)](#)

firstChild(hierarchy, location)

firstChild(hierarchy, location), given a hierarchy and a location, returns the first child. The function finds the parent location member along the hierarchy and then finds the first child, who is a sibling of the location member passed as an argument.

firstChild(Location location, Hierarchy h)

firstChild(hierarchy, member)

`firstChild(hierarchy, member)`, given a hierarchy and a member, returns the first child. The function first finds the parent of the member along the hierarchy and then finds the first child, who is a sibling of the member passed as an argument.

```
firstChild(Member member, Hierarchy h)
```

firstChild(hierarchy)

`firstChild(hierarchy)`, given a hierarchy and the current location in the specified context, returns the first child. the function first finds the parent of the current-location member along the hierarchy and then finds the first child, who is a sibling of the current-location member.

```
firstChild(Hierarchy h)
```

firstDescendant

`firstDescendant`, given a level and a member, returns the first child of the member at the given level. If this member is also at this level, the function returns itself; otherwise, the function returns the first descendant at this level.

```
firstDescendant(Member member, Level l)
```

hierarchy

`hierarchy` casts a hierarchy into a hierarchy. Used to resolve ambiguity resulting from two or more function signatures with the same name. For example, assume that you have two function signatures. One takes a hierarchy and the other takes a dimension. If the name of the hierarchy and the dimension are the same, then invoking that function in a formula creates ambiguity. By explicitly casting the string with the `hierarchy` function, the ambiguity is resolved.

```
hierarchy(Hierarchy h)
```

hierarchyLevel

`hierarchyLevel` casts a level into a level. Used to resolve ambiguity resulting from function signatures with the same name. For example, assume that you have two function signatures, and one takes a level and the other takes a hierarchy. If the name of the level and the hierarchy are the same, then invoking that function in a formula creates ambiguity. By explicitly casting the string with the `hierarchyLevel` function, the ambiguity is resolved.

```
hierarchyLevel(levelObj)
```

in

`in` returns true if the member is in the set.

```
in(Member member, Member[] set)
```

indexOf

`indexOf` returns an index of the specified location along the specified dimension.

indexOf(Location location, Hierarchy h)

is Functions

Subtopics

- [isAncestor](#)
- [isBatchMode](#)
- [isInRange](#)
- [isLeaf](#)
- [isLevel](#)
- [isLevelIn](#)
- [isMember](#)
- [isRoot](#)

isAncestor

`isAncestor` checks whether the dependency relationship between ancestor and descendant holds.

isAncestor(Member ancestor, Member descendant)

isBatchMode

`isBatchMode` checks whether the current sandbox is in batch mode. Used in `if` statements in formulas to define whether to perform certain operations in batch or interactive mode.

isBatchMode()

isInRange

`isInRange` returns true if the supplied location is in a valid range.

isInRange(Location location)

isLeaf

`isLeaf` returns true if the coordinate of the location in context along the given dimension is a leaf member.

isLeaf(Hierarchy h)

isLevel

`isLevel` returns true if the coordinate of the location in context is along the supplied level

isLevel(Level level)

isLeveln

isLeveln returns true if the coordinate of the location in context along the supplied dimension is at the supplied level number(leveln)

isLeveln(Hierarchy h, double leveln)

isMember

isMember returns true if the coordinate of the location in context along the given dimension is a member whose name is the same as the given name

isMember(Dimension d, String name)

isRoot

isRoot(Member m, Hierarchy h)

isRoot(member, hierarchy) returns true if the specified member along the given hierarchy is a root member.

isRoot(Member m, Hierarchy h)

isRoot(member)

isRoot(member) returns true if the specified member is a root member along the default hierarchy of its dimension

isRoot(Member m)

lastChild

Subtopics

- [lastChild\(hierarchy\)](#)
- [lastChild\(location, hierarchy\)](#)
- [lastChild\(member, hierarchy\)](#)

lastChild(hierarchy)

lastChild(hierarchy), given the current context, returns the member that belongs to the hierarchy and returns the last child of that member in the hierarchy

lastChild(Hierarchy h)

lastChild(location, hierarchy)

`lastChild(location, hierarchy)`, given the hierarchy and the location, returns the member that belongs to the hierarchy from the given location and returns the last child of that member in the hierarchy

lastChild(Location location, Hierarchy h)

lastChild(member, hierarchy)

`lastChild(member, hierarchy)`, given the hierarchy and the member, returns the last child of the member in the hierarchy

lastChild(Member member, Hierarchy h)

lastDescendant

`lastDescendant`, given a level and a member, returns the last child of the member at the given level. If this member is also at this level, the function returns itself; otherwise, the function returns the last descendant at this level.

lastDescendant(Member member, Level l)

location

`location` casts a location into a location. Used to resolve ambiguity resulting from function signatures with the same name. For example, assume you have two function signatures, and one takes a hierarchy and the other takes a measure. If the name of the hierarchy and the measure are the same, then invoking that function in a formula creates ambiguity. By explicitly casting the string with the `location` function, the ambiguity is resolved.

location(loc)

lookup functions

Subtopics

- [lookup](#)
- [lookupDate](#)
- [lookupString](#)

lookup

Subtopics

- [lookup\(one key column\)](#)
- [lookup\(two key columns\)](#)
- [lookup\(three key columns\)](#)
- [lookup\(four key columns\)](#)

lookup(one key column)

`lookup(one key column)`, given a key column, constructs the row source key, looks up the row source entry, and returns the double numeric value of `resultColumn` from the row source.

lookup(String rowsourceName, String keyColumn1, String resultColumn)

See [“lookup” on page 211](#).

lookup(two key columns)

`lookup(two key columns)` looks up a row source with two key-columns for the value of `resultColumn`.

lookup(String rowsourceName, String keyColumn1, String keyColumn2, String resultColumn)

See [“lookup” on page 211](#).

lookup(three key columns)

`lookup(three key columns)` looks up a row source with three key-columns for the value of `resultColumn`.

lookup(String rowsourceName, String keyColumn1, String keyColumn2, String keyColumn3, String resultColumn)

See [“lookup” on page 211](#).

lookup(four key columns)

`lookup(four key columns)` looks up a row source with four key-columns for the value of `resultColumn`

lookup(String rowsourceName, String keyColumn1, String keyColumn2, String keyColumn3, String keyColumn4, String resultColumn)

lookupDate

Subtopics

- `lookupDate(one key column)`
- `lookupDate(two key columns)`
- `lookupDate(three key columns)`
- `lookupDate(four key columns)`

lookupDate(one key column)

`lookupDate(one key column)`, given a key column, constructs the row source key, looks up the row source entry, and returns the date value of `resultColumn` from the row source.

lookupDate(String rowsourceName, String keyColumn1, String resultColumn)

lookupDate(two key columns)

`lookupDate(two key columns)` looks up a row source with two key-columns for the value `resultColumn`

lookupDate(String rowsourceName, String keyColumn1, String keyColumn2, String resultColumn)

lookupDate(three key columns)

`lookupDate(three key columns)` looks up a row source with three key-columns for the value `resultColumn`

lookupDate(String rowsourceName, String keyColumn1, String keyColumn2, String keyColumn3, String resultColumn)

lookupDate(four key columns)

`lookupDate(four key columns)` looks up a row source with four key-columns for the value `resultColumn`

lookupDate(String rowsourceName, String keyColumn1, String keyColumn2, String keyColumn3, String keyColumn4, String resultColumn)

lookupString

Subtopics

- `lookupString(one key column)`
- `lookupString(two key columns)`
- `lookupString(three key columns)`
- `lookupString(four key columns)`

lookupString(one key column)

`lookupString(one key column)`, given a key column, constructs the row source key, looks up the row source entry, and returns the string value of `resultColumn` from the row source.

lookupString(String rowsourceName, String keyColumn1, String resultColumn)

lookupString(two key columns)

`lookupString(two key columns)` looks up a row source with two key-columns for the value `resultColumn`

lookup(String rowsourceName, String keyColumn1, String keyColumn2, String resultColumn)

lookupString(three key columns)

`lookupString(three key columns)` looks up a row source with three key-columns for the value `resultColumn`

lookup(String rowsourceName, String keyColumn1, String keyColumn2, String keyColumn3, String resultColumn)

lookupString(four key columns)

`lookupString(four key columns)` looks up a row source with four key-columns for the value `resultColumn`

lookupString(String rowsourceName, String keyColumn1, String keyColumn2, String keyColumn3, String keyColumn4, String resultColumn)

member

Subtopics

- `member(dimension)`
- `member(location, dimension)`

member(dimension)

`member(dimension)`, given a location and a dimension, returns the member in the location that belongs to the given dimension

member(Dimension d)

member(location, dimension)

`member(location, dimension)`, given the current context and a dimension, returns the member in the current context that belongs to the given dimension

member(Location location, Dimension d)

memberName

Subtopics

- `memberName(location, dimension)`
- `memberName(member)`
- `memberName(dimension)`

memberName(location, dimension)

`memberName(location, dimension)`, given a location and a dimension, returns the name of the member in the location at dimension “d.” The dimension can be dense (represented by location) or sparse (represented by the context in which the location exists).

memberName(Location location, Dimension d)

memberName(member)

`memberName(member)` returns the name of the specified member

memberName(Member m)

memberName(dimension)

`memberName(dimension)` given the current context, finds the member representing the given dimension and returns the name of that member.

memberName(Dimension d)

nextMember

Subtopics

- `nextMember(level)`
- `nextMember(member, level)`

nextMember(level)

`nextMember(level)` returns the next member in the specified hierarchy

nextMember(Level level)

nextMember(member, level)

`nextMember(member, level)` returns the next member in the specified level of the hierarchy for the specified member.

nextMember(Member m, Level level)

memberProperty

`memberProperty` returns the value of attribute with the specified name from the specified member. Returns null if the member is null or if the attribute is not found. Predefined attributes are name, description, display name, share, name space, qualified name, qualified display name, start date, end date, number of days, and actual plan.

memberProperty(Member m, String attributeName)

parent

Subtopics

- `parent(hierarchy)`
- `parent(location, hierarchy)`
- `parent(member, hierarchy)`

parent(hierarchy)

`parent(hierarchy)` returns the parent location along the dimension of the specified location

parent(Hierarchy h)

parent(location, hierarchy)

`parent(location, hierarchy)` returns the parent location along the dimension of the specified location

parent(Location loc, Hierarchy h)

parent(member, hierarchy)

parent(member, hierarchy) returns the parent location along the dimension of the specified location

parent (Member m, Hierarchy h)

parentMembers

parentMembers, given a member and a dimension, returns all parent members along the dimension (Parents are defined in terms of calculation dependencies.)

parentMembers (<ArrayList<Member> current, ArrayList<String> extraParams, Dimension d, String namespace)

parents

parents, given a member and a dimension, returns all parent members along the dimension (Parents are defined in terms of calculation dependencies.)

parents (Member child, Dimension d, String namespace)

pastCousinLocations

Subtopics

- [pastCousinLocations\(ancestor member\)](#)
- [pastCousinLocations\(root member\)](#)

pastCousinLocations(ancestor member)

pastCousinLocations(ancestor member), given an ancestor member, a location, and a hierarchy for a dimension, the cousin location is where the member in the dimension of the specified location is replaced with members that are descendants of the **ancestor** member but at the same level as the member in the specified location. It is assumed that the dimension of the ancestor member is same as that of the specified hierarchy. The locations are ordered from left to right along the level of the specified location. The boolean flags control which locations to ignore (to the left or to the right). This method returns only locations that have time members in the past relative to the current system time.

pastCousinLocations (Location loc, Member ancestor, Hierarchy h, boolean excludeLeftCousins, boolean excludeRightCousins, boolean excludeSelf)

pastCousinLocations(root member)

pastCousinLocations(root member), given an “root” member (implicit ancestor), a location, and a hierarchy for a dimension, the cousin location is where the member in the dimension of the specified location is replaced with members that are descendants of the

ancestor member but at the same level as the member in the specified location. It is assumed that the dimension of the ancestor member is same as that of the specified hierarchy. The locations are ordered from left to right along the level of the specified location. The boolean flags control which locations to ignore (to the left or to the right). This method returns only locations that have time members in the past relative to the current system time.

pastCousinLocations(Location loc, Hierarchy h, boolean excludeLeftCousins, boolean excludeRightCousins, boolean excludeSelf)

previousMember

Subtopics

- `previousMember(Level level)`
- `previousMember(member, level)`

previousMember(Level level)

`previousMember(level)` returns the previous member in the specified hierarchy.

previousMember(Level level)

previousMember(member, level)

`previousMember(member, level)` returns the previous member in the specified level of the hierarchy for the specified member.

previousMember(Member m, Level level)

property

`property` returns the value of the attribute with specified name from the member in the current context along the specified dimension. Returns null if the member is null or if the attribute is not found. Predefined attributes are name, description, display name, share, namespace, qualified name, qualified display name, start date, end date, number of days, and actual plan.

property(Dimension d, String attributeName)

range

Subtopics

- `range(from an anchor member)`
- `range(member from, member to)`
- `range(member from, member to, Level level)`

range(from an anchor member)

`range` returns an array of members on the specified level from the member that has distance start from the anchor member to the member that has distance end from the anchor member.

range(Member anchor, Level level, double start, double end)

range(member from, member to)

`range` returns an array of members on the specified level between the **from** member and the **to** member. The inclusion of end members is controlled by the boolean flags.

range(Member from, boolean includeFrom, Member to, boolean includeTo, Level level)

range(member from, member to, Level level)

`range(Member from, Member to, Level level)` returns an array of members on the specified level between the **from** member and the **to** member. The end members are included in the result.

range(Member from, Member to, Level level)

siblingCount

Subtopics

- `siblingCount(heirarchy)`
- `siblingCount(location, hierarchy)`

siblingCount(heirarchy)

`siblingCount(heirarchy)` returns the number of siblings in the location supplied in the context map along the supplied dimension.

siblingCount(Hierarchy h)

siblingCount(location, hierarchy)

`siblingCount(location, hierarchy)` returns the number of siblings in the location along the specified hierarchy. Siblings are defined here as the children of the same parent in the specified hierarchy that includes the given location.

siblingCount(Location location, Hierarchy h)

siblingLocations

Subtopics

- `siblingLocations(hierarchy)`
- `siblingLocations(location, hierarchy)`
- `siblingLocations(location, hierarchy, boolean)`

siblingLocations(hierarchy)

`siblingLocations(hierarchy)` returns the list of siblings in the location specified in the context map along the specified hierarchy.

siblingLocations(Hierarchy h)

siblingLocations(location, hierarchy)

`siblingLocations(location, hierarchy)` returns the list of siblings in the specified location along the specified hierarchy.

siblingLocations(Location loc, Hierarchy h)

siblingLocations(location, hierarchy, boolean)

`siblingLocations(location, hierarchy, boolean)` returns the list of siblings in the specified location along the specified hierarchy. The boolean flags control which siblings to exclude in the output.

siblingLocations(Location loc, Hierarchy h, boolean excludeLeftSiblings, boolean excludeRightSiblings, boolean excludeSelf)

Shape Functions

Subtopics

- `all`
- `current`
- `first`
- `lag`
- `last`
- `lead`
- `level`
- `leveln`
- `levels`
- `moduloNext`
- `moduloPrevious`
- `next`
- `nonLeaves`
- `offset`
- `previous`
- `range`
- `siblings`
- `singleMember`

Shape functions manipulate the shape of the slices where the formula compilation happens or where the calculation happens. For the current context being parsed, there is an associated sparse shape and dense shape. The scope is narrowed by calling in the functions. Shape functions are also used in measure formulas to limit or describe the scope of the computation. They define the scope of the source and target slices in the allocation map and the cube map.

`all`

`all` adds all members from the supplied dimension to the calculation shape.

`all`(Dimension d)

`current`

`current` returns the time dimension in the current context. The current time member is then added to the calculation shape.

`current` ()

`first`

`first` adds the location that is the same as the context except for the index along the supplied dimension (the context member's left-most sibling).

first(Hierarchy h)

lag

lag adds the location that is “lagBy” time periods along the specified hierarchy behind the current member in the calculation shape.

lag(Hierarchy h, double lagBy)

last

last adds the location that is the same as the context except for the index along the supplied dimension (the context member's right-most sibling).

last(Hierarchy h)

lead

lead adds the location that is “leadBy” time periods along the specified hierarchy ahead of the current member in the calculation shape.

lead(Hierarchy h, double leadBy)

level

Subtopics

- [level\(level, one attribute\)](#)
- [level\(level, one attribute\)](#)
- [level\(level\)](#)

level(level, one attribute)

level(level, one attribute) for the current context being parsed, filter to the specified level for the associated sparse or dense shape. In the specified level, filter the members by the specified attribute value.

level(Level level, String attributeName, String attributeValue)

level(level, one attribute)

level(level, two attributes) for the current context being parsed, filter to the specified level for the associated sparse or dense shape. In the specified level, filter the members by the specified attribute values. Only the members that match both attribute values are included in the scope.

level(Level level, String attributeName, String attributeValue, String attributeName1, String attributeValue1)

level(level)

level(level) for the current context being parsed, filter to the specified level for the associated sparse or dense shape.

level(Level level)

leveln

leveln, given a dimension and a level number, adds the set of members from that level to the supplied odometer.

leveln(Dimension d, String hierName, Double levelNumber)

levels

levels, for the current context being parsed, filter to the specified level for the associated sparse or dense shape. You can filter up to seven levels.

levels(Level level1, Level level2, Level level3, Level level4, Level level5, Level level6, Level level7)

moduloNext

moduloNext, given a dimension and a location, adds the next location to the calculation shape.

moduloNext(Dimension d)

moduloPrevious

moduloPrevious, given a dimension and a location, adds the previous location to the calculation shape.

moduloPrevious(Dimension d)

next

next, given a dimension and a location, adds the next location to the calculation shape.

next(Dimension d)

nonLeaves

nonLeaves, adds all members at the non-leaf level (has a child) from the supplied dimension to the calculation shape.

nonLeaves(Dimension d)

offset

`offset`, given a dimension and a location, adds the location at the relative offset in the same level.

offset(Hierarchy h, double offset)

previous

`previous`, given a dimension and a location, adds the previous location to the calculation shape.

previous(Dimension d)

range

Subtopics

- `range(from, to)`
- `range(zero, to)`

range(from, to)

`range (from, to)`, populates a collection of locations along the supplied dimension starting from the offset “from” (relative the current location) to the offset “to” (relative to the current location).

range(Hierarchy h, double from, double to)

range(zero, to)

`range(zero, to) (zero, to)`, populates a collection of locations along the supplied dimension starting from offset zero (relative to the current location) to the offset **to** (relative to the current location).

range(Hierarchy h, double to)

siblings

Subtopics

- `siblings(hierarchy)`
- `siblings(location, hierarchy)`

siblings(hierarchy)

`siblings (hierarchy)`, adds the siblings in the location specified in the context map along the supplied dimension in the calculation shape.

siblings(Hierarchy h)

siblings(location, hierarchy)

`siblings(location, hierarchy)`, adds the siblings in the location specified in the context map along the supplied dimension in the calculation shape.

siblings(Location loc, Hierarchy h)

singleMember

`singleMember` adds a single member from the supplied dimension to the calculation shape.

singleMember(Dimension d, String mbr).

Consider the following example:

```
"Beginning Inventory Units" = "Ending Inventory Units"[previous(Manufacturing)]
Required[level(Week)] =
    if(isNull("Schedule Required"),
        "MRP Receipts"[lead(Manufacturing, ComponentMetrics.leadTime / 7)],
        "Schedule Required")
"Demand Units Avg"[level(Week)] = avg("Demand Units"[sibling(Manufacturing)])
```

In this example:

- The shape functions, `previous`, `level`, `lead`, and `sibling`:
 - Are used with a location operator, and share some commonality in their signature and implementation
 - Return a Dimension instance that tells the location operator on which dimension to qualify
 - Are evaluated at compile-time and at runtime.
- `"MRP Receipts"[lead(Manufacturing, ComponentMetrics.leadTime / 7)]` calculates the lead time value only at runtime. Even though the lead time value is not known until runtime, it is crucial to know the cell locations on which the measure `Required` is based so that Integrated Operational Planning can calculate `MRP Receipts` before calculating `Required`.

The following shape function example calculates the leaf-level members of a dimension.

Returning the dimension at the end of the call is used to generate the location collections.

The following example shows how to handle parameters whose values are not determined at compile time.

```
public static Dimension leaves(Dimension d) throws ISEException
{
    VariableShape varShape = ctx.getVariableShape();
    CalcOdometer denseShape = varShape.getDenseShape();
    Slice sparseShape = varShape.getSparseShape();

    DimensionType dtype = d.getType();
    if (dtype.equals(DimensionType.SPARSE)) {
        if (sparseShape == null)
            throw new ISEException("populating for sparse dimension, "+d.getName()+
                                   " is not allowed");
    }
}
```



```

        MemberSet mset = sparseShape.locate(d);
        Iterator mbrs = d.getMemberAccessor().getMemberIterator();
        while (mbrs.hasNext()) {
            Member m = (Member)mbrs.next();
            if (!d.hasDependents(m)) {
                mset.add(m);
            }
        }
    }
    else {
        Iterator mbrs = d.getMemberAccessor().getMemberIterator();
        while (mbrs.hasNext()) {
            Member m = (Member)mbrs.next();
            if (!d.hasDependents(m)) {
                denseShape.addMember(m);
            }
        }
    }
}

return d;
}

public static Dimension lead(Hierarchy h, double leadBy)
    throws ISEException
{
    Location current = ctx.getVariableShape().getCurrentLocation();

    int offset = (int)Math.round(leadBy);
    Member [] mbrs = current.getMembers();
    Dimension d = h.getDimension();
    int idx = Location.getDimIndex(mbrs, d);
    Member startMbr = mbrs[idx];
    HierarchyNode node = h.getHierarchyNode(startMbr);

    Level[] levels = h.getLevels();
    int levelIndex = node.getLevel();
    long numOfMembersAtLowerLevels = 0;
    for (int i = levels.length-1; i > levelIndex; i--) {
        numOfMembersAtLowerLevels += levels[i].getMemberCount();
    }

    long membersAtThisLevel = levels[levelIndex].getMemberCount();

    MemberSet sparseMemberSet = null;
    CalcOdometer denseShape = null;
    Member offsetMember = null;

    if (d.getType().equals(DimensionType.SPARSE)) {
        Slice sparseShape = ctx.getVariableShape().getSparseShape();
        sparseMemberSet = sparseShape.locate(d);
    }
    else {
        denseShape = ctx.getVariableShape().getDenseShape();
    }

    if (ctx.isAtCompileTime() && ctx.hasDynamicParameters()) {
        long lowerBound = numOfMembersAtLowerLevels;
        long upperBound = numOfMembersAtLowerLevels + membersAtThisLevel - 1;
    }
}

```

```

    long currentDepID = startMbr.getDependencyID();

    if (goForward) {
        for (long k = currentDepID; k <= upperBound; k++) {
            offsetMember = d.findMemberByDependencyID(k);
            if (offsetMember != null) {
                if (sparseMemberSet != null)
                    sparseMemberSet.add(offsetMember);
                else if (denseShape != null)
                    denseShape.addMember(offsetMember);
            }
        }
    }
    else {
        for (long k = lowerBound; k <= currentDepID; k++) {
            offsetMember = d.findMemberByDependencyID(k);
            if (offsetMember != null) {
                if (sparseMemberSet != null)
                    sparseMemberSet.add(offsetMember);
                else if (denseShape != null)
                    denseShape.addMember(offsetMember);
            }
        }
    }

    return d;
}

if (offset == 0) {
    offsetMember = startMbr;
}
else {
    long dependentID = startMbr.getDependencyID()+offset;

    boolean isvalid = true;
    if (dependentID < numOfMembersAtLowerLevels)
    {
        dependentID = numOfMembersAtLowerLevels;
        isvalid = false;
    }
    else
    if (dependentID >= numOfMembersAtLowerLevels + membersAtThisLevel)
    {
        dependentID = numOfMembersAtLowerLevels + membersAtThisLevel - 1;
        isvalid = false;
    }

    if (isvalid)
        offsetMember = d.findMemberByDependencyID(dependentID);
}

if (offsetMember != null) {
    if (sparseMemberSet != null)
        sparseMemberSet.add(offsetMember);
    else if (denseShape != null)
        denseShape.addMember(offsetMember);
}

```

```

    }
    return d;
}

```

Time Functions

Subtopics

- `currentTime`
- `currentTimeMember`
- `futureSiblingsCount`
- `getDateForCurrentTimeMember`
- `isCurrent`
- `isCurrentDateInRange`
- `isEndOfPeriod`
- `isFuture`
- `isInRange`
- `isInRangeFromCurrentTime`
- `isLeftRightLeaf`
- `isNext`
- `isPast`
- `isPrevious`
- `isStartOfPeriod`
- `offsetFromCurrent`
- `overlapMembers`
- `pastSiblingsCount`
- `timeRange`

Time functions query time dimension members for associated time properties. Each time member has an associated start time and end time. At anytime, a current member's time span includes the current “wall-clock time.” Past and future members are defined in relation to the current member. Unlike other dimension members, time members are ordered from earliest to latest. Time functions are used in formulas for computing derived measures.

currentTime

`currentTime` returns the time member for the current system time in the specified level.

currentTime(Level level)

currentTimeMember

`currentTimeMember` finds the time member in the current location in the program.

currentTimeMember()

futureSiblingsCount

`futureSiblingsCount`, given a time-member at a level, finds the number of siblings that are in the future (children of the same parent). For example, you could use `futureSiblingsCount` to calculate the number of weeks remaining in a month.

futureSiblingsCount (Member member)

getDateForCurrentTimeMember

`getDateForCurrentTimeMember` returns the starting time of the time member in the current context.

getDateForCurrentTimeMember ()

isCurrent

Subtopics

- `isCurrent(current location)`
- `isCurrent(member)`
- `isCurrent(supplied location)`

isCurrent(current location)

`isCurrent(current location)` returns true if the time bit of the current location includes the current server time.

isCurrent ()

isCurrent(member)

`isCurrent(member)` returns true if the time bit of the supplied member includes the current server time.

isCurrent (Member timeMember)

isCurrent(supplied location)

`isCurrent(supplied location)` returns true if the time bit of the supplied location is the same as the current time period.

isCurrent (Location location)

isCurrentDateInRange

`isCurrentDateInRange` checks whether the date is in range between `startDate` and `endDate` and returns “true” if in range and returns “false” if not in range.

isCurrentDateInRange(Date startDate, Date endDate)

isEndofPeriod

isEndofPeriod returns true if the current member is at the start of supplied period.

isEndOfPeriod(String period)

isFuture

Subtopics

- [isFuture\(current location\)](#)
- [isFuture\(member\)](#)
- [isFuture\(supplied location\)](#)

isFuture(current location)

isFuture(current location) returns true if the time bit of the current location is in the future.

isFuture()

isFuture(member)

isFuture(member) returns true if the time bit of the supplied member is in the future.

isFuture(Member timeMember)

isFuture(supplied location)

isFuture(supplied location) returns true if the time bit of the supplied location is in the future.

isFuture(Location location)

isInRange

isInRange checks whether the *query* date is in range between startDate and endDate and returns “true” if it is in range and returns “false” if it is not in range.

isInRange(Date startDate, Date endDate, Date query)

isInRangeFromCurrentTime

isInRangeFromCurrentTime checks whether the time member of the current cell location is within the given range from the current time member (based on server start time). If the offset

is positive, then checks whether the cell location is in the future of “current time” within the specified range. If the offset is negative, then checks whether it is in the past within a given range.

isInRangeFromCurrentTime(Location cellLocation, Dimension d, double rangeLimit)

where cellLocation is the location being examined, d is the time dimension of interest, and rangeLimit is the range limit in terms of the number of time units(weeks, months, and so on.)

isLeftRightLeaf

isLeftRightLeaf returns true if the node is in the left- or right-most leaf of the subtree rooted with ancestor of node “n” that belongs to the supplied level.

isLeftRightLeaf(Level level, HierarchyNode n, boolean left)

isNext

Subtopics

- `isNext()`
- `isNext(member)`
- `isNext(supplied location)`

isNext()

isNext() returns true if the supplied time member is the next time member at the same level.

isNext()

isNext(member)

isNext(member) returns true if the supplied location is the next time member at the same level.

isNext(Member timeMember)

isNext(supplied location)

isNext(supplied location) returns true if the supplied location is the next time member at the same level.

isNext(Location location)

isPast

Subtopics

- `isPast(member)`
- `isPast(current location)`
- `isPast(supplied location)`

isPast(member)

`isPast(member)` returns true if the supplied time member is in the past.

isPast(Member timeMember)

isPast(current location)

`isPast(current location)` returns true if the time bit of current location is in the past.

isPast()

isPast(supplied location)

`isPast(supplied location)` returns true if the time bit of the supplied location is in the past.

isPast(Location location)

isPrevious

Subtopics

- `isPrevious(member)`
- `isPrevious()`
- `isPrevious(supplied location)`

isPrevious(member)

`isPrevious(member)` returns true if the supplied time member is the previous time member at the same level.

isPrevious(Member timeMember)

isPrevious()

`isPrevious()` returns true if the supplied time member is the previous time member at the same level.

isPrevious()

isPrevious(supplied location)

`isPrevious (supplied location)` returns true if the supplied location is the previous time member at the same level.

isPrevious(Location location)

isStartOfPeriod

`isStartOfPeriod` returns true if the current member is at the end of the supplied period.

isStartOfPeriod(String period)

offsetFromCurrent

`offsetFromCurrent`, given a current location, finds the offset in units from the current time member. Assumes cell locations are at the leaf level. Not supported for non-leaf levels. Returns a positive number for future time periods, a negative number for past time periods, and zero for the current time period.

offsetFromCurrent ()

overlapMembers

`overlapMembers`, given a source level and a target member in a time dimension, find all the members in the source level that intersect with the target member. (The source level and target member do not need to belong to the same dimension.)

overlapMembers(Level sourceLevel, Member targetMember)

pastSiblingsCount

`pastSiblingsCount`, given a time member at a level, finds the number of siblings that are in the past (children of the same parent). For example, you could use `pastSiblingsCount` to calculate the number of previous weeks in a month.

pastSiblingsCount(Member member)

timeRange

`timeRange` returns true if the time member for the current system time is for that level.

timeRange(Level level, double offset)

External Functions

External functions can be implemented by yourself or by a third party.

Following is an example of an external function:

```
java.lang.StrictMath.abs("Ending Inventory Units")
```

In this function:

- If you imported the Math class by the system properties, you would have:

```
StrictMath.abs("Ending Inventory Units")
```

- If the function abs is unique, you would have:

```
abs("Ending Inventory Units")
```

Custom Functions

The formula compiler detects evaluation dependency by identifying the places where value of a cell variable is retrieved. For example, with the following formula, you need to compute “Scrap Units” before you compute “Scrap Cost”

```
"Scrap Cost" = ComponentMetrics.standardCost * "Scrap Units"
```

However, if a function takes a parameter of Location type, the compiler does not know whether the function implementation takes the metadata of the location only, or it takes its value also. The protocol described as following serves such a purpose.

If a custom function reads values from some cell locations rc1, rc2, ..., rcm, from the current block and writes into some cell locations, wc1, wc2, ..., wcn, in the current block, you need to know that during evaluation dependency analysis. In this case, you need to have rc1, rc2, ..., rcm evaluated first before this customer function is called. Create a HashMap instance, which has each rci as key and a HashSet instance with wc1, ..., wcn as the value.



MDX Extensions

In This Appendix

<code>Reverse(set)</code>	235
<code>RSQLGenerateSet(RSQLQuery [,dimHierarchy, NameColumn,NameSpaceColumn]+)</code>	235
<code>RowSourceLookup(RowSourceColumn [, tuple])</code>	236
<code>SystemPeriod(level)</code>	236

The following functions expose Integrated Operational Planning functionality in MDX.

Reverse(*set*)

`Reverse(set)` returns the set reversed from the input set

Example:

```
Reverse({ [January] , [February] , [March] })
```

returns:

```
{ [March] , [February] , [January] }
```

RSQLGenerateSet(*RSQLQuery* *[,dimHierarchy,* *NameColumn,NameSpaceColumn]+)*

`RSQLGenerateSet(RSQLQuery [,dimHierarchy, NameColumn,NameSpaceColumn]+)`
returns the defined set

Example:

```
RSQLGenerateSet("SELECT model from RoleSecurity where permission = 'E' and user =  
'admin'",[Product], "model", "")
```

returns a set of products (product dimension members) with specific permission for the administrator user in the `RoleSecurity` row source.

RowSourceLookup (*RowSourceColumn* [, tuple])

RowSourceLookup (*RowSourceColumn* [, tuple]) returns the row source column value for the current cell. *RowSourceColumn* is specified as an Identifier with the form [*RowSource Name*] . [*RowSource Column Name*] . The return value type is the same as the row source column referenced.

Example:

```
RowSourceLookup ( [ComponentMetrics] . [standardCost] )
```

returns the value standard cost from the current cell's ComponentMetrics row source.

SystemPeriod (*level*)

SystemPeriod (*level*) returns the member from the specified level of the Time dimension. The returned member (date/time span) contains the current date and time at the time of execution.

Example:

```
SystemPeriod ( [Fiscal] . [Quarter] )
```

returns the member from the quarter level of the Time (Fiscal) dimension containing the current date and time (the current quarter).



Load XML Specifications

In This Appendix

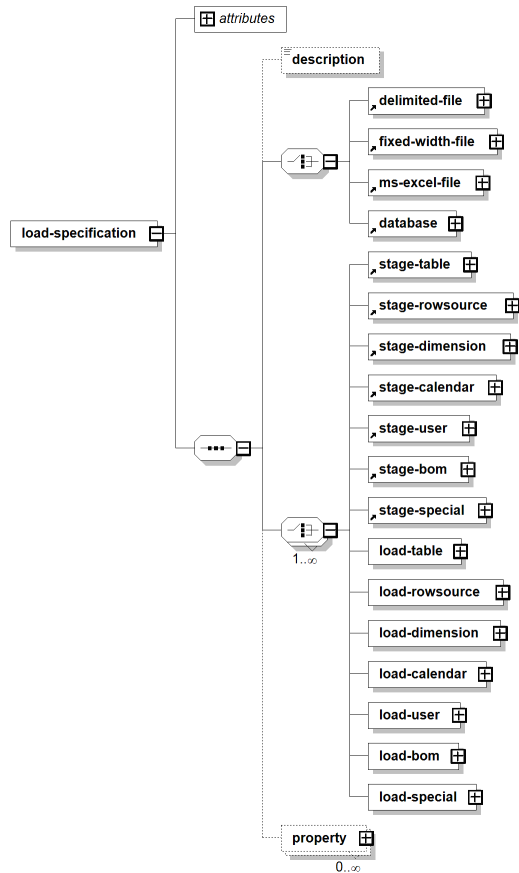
About Load XML Specifications	237
Elements	240
Value Functions	269
Properties	279

About Load XML Specifications

This appendix describes the load XML specifications. You can use these load specification mappings in a stagemap.

Each load specification contains attributes; a description; one of these elements: <delimited-file>, <fixed-width-file>, <ms-excel-file>, or <database>; and optionally, one or more of the <stage...> or <load...> elements. See [Figure 12](#).

Figure 12 Load Specification Contents



The following is a sample Load Forecast Demand XML file:

```

<load-specification xmlns="http://schemas.interlacedsystems.com/3.0/load-specification"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.interlacedsystems.com/3.0/
  load-specification loadspec-3.0.xsd">

  <delimited-file start-row="1" delimiter="," text-qualifier=""">
    <field index="2" name="product" />
    <field index="3" name="measure" />
    <field index="4" name="startDate" />
    <field index="5" name="offset" type="int" />
    <field index="7" name="forecast" type="double" />
  </delimited-file>

  <stage-rowsource rowsource="ForecastPro">
    <simple-filter field="{measure}" value="Booked Units" />
    <map column="product" value="{product}" />
    <map column="measure" value="Statistical Units" />
    <map column="forecastDate" value="$beginmember({startDate},
    {offset})" />
    <map column="quantity" value="{forecast}" />
  </stage-rowsource>

  <stage-rowsource rowsource="ForecastPro">
    <simple-filter field="{measure}" value="Booked Sales" />
  </stage-rowsource>

```

```

    <map column="product" value="{product}" />
    <map column="measure" value="Statistical Sales" />
    <map column="forecastDate" value="$beginmember({startDate},
    ${offset})" />
    <map column="revenue" value="{forecast}" />
  </stage-rowsource>

```

```

  <property name="batch.size" value="1" />

```

```

</load-specification>

```

The following is a sample Load RMO Capacity XML file:

```

<load-specification xmlns="http://schemas.interlacedsystems.com/3.0/load-specification"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.interlacedsystems.com/3.0/
  load-specification loadspec-3.0.xsd">

  <ms-excel-file sheet="Media Factory Capacity" start-row="4"
  row-increment="1" skip-hidden-rows="true">
    <field column="A" name="SUPPLY_SITE" nullable="false" />
    <field column="B" name="C250B_200L" nullable="false" />
    <pivot row="2" columns="C:M" name="TIME_MEMBER" nullable="true"
    default-to-previous-non-null="false" />
    <pivot row="3" columns="C:M" format="MM/dd/yyyy" name="START_DATE"
    type="date" />
    <pivot columns="C:M" name="CAPACITY" type="double" nullable="true"
    default-to-previous-non-null="false" />
  </ms-excel-file>

  <stage-table table="IN_STG_RMO_CAPACITY">
    <map column="SUPPLY_SITE" value="{SUPPLY_SITE}" />
    <map column="C250B_200L" value="{C250B_200L}" />
    <map column="START_DATE" value="{START_DATE}" />
    <map column="TIME_MEMBER" value="{TIME_MEMBER}" />
    <map column="CAPACITY" value="{CAPACITY}" />
  </stage-table>
  <property name="batch.size" value="100" />
</load-specification>

```

Elements

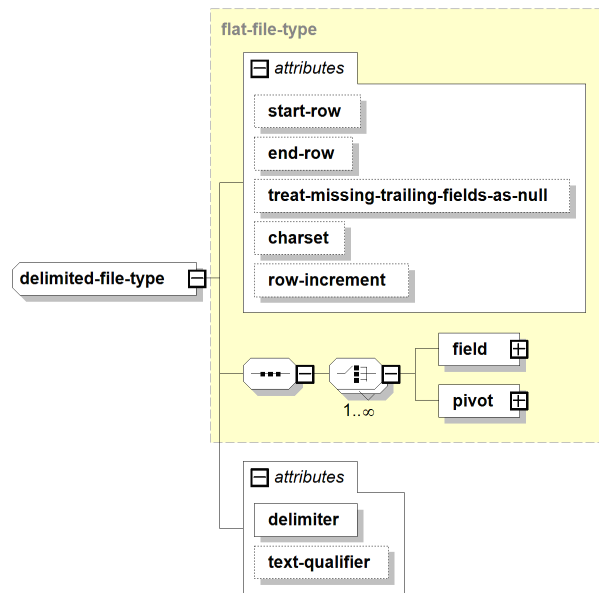
Subtopics

- [<delimited-file>](#)
- [<fixed-width-file>](#)
- [<ms-excel-file>](#)
- [<database>](#)
- [<database> / <query>](#)
- [<flat-file> / <field>](#)
- [<ms-excel-file> / <field>](#)
- [<field> / <translate>](#)
- [<field> / <range>](#)
- [<flat-file> / <pivot>](#)
- [<excel-file> / <pivot>](#)
- [<stage-dimension>](#)
- [<stage-dimension> / <member>](#)
- [<stage-dimension> / <member> / <map>](#)
- [<stage-calendar>](#)
- [<stage-rowsource>](#)
- [<stage-rowsource> / <map>](#)
- [<stage-bom>](#)
- [<stage-bom> / <map>](#)
- [<stage-user>](#)
- [<stage-user> / <map>](#)
- [<stage-table>](#)
- [<stage-table> / <map>](#)
- [<stage-special>](#)
- [<stage-special> / <param>](#)
- [<prologue> | <epilogue>](#)
- [\(<prologue> | <epilogue>\) / <param>](#)
- [<simple-filter>](#)
- [<dimension-filter>](#)
- [<rowsource-filter>](#)
- [<distinct-filter>](#)
- [<custom-filter>](#)

This section describes elements that you can include in a load specification.

<delimited-file>

Fields in the data file are separated by a delimiter character.



start-row

Row from which to start loading. For example, `start-row="2 "` tells the loader to skip the first row. (Default=1)

end-row

Row after which to stop loading. (Default=end of file or first empty row)

treat-missing-trailing-fields-as-null

Whether to treat missing trailing fields (files with fewer fields than defined in the load specification) as null values instead of errors. (Default=false)

charset

Character set used by the data file. Generally defined only for files that use a character set different from that of the computer on which the loader is running. (By default, the character set is not specified.)

row-increment

Number of rows between each record in the file. Generally defined only for files containing multiple data pivots. (Default=1)

delimiter

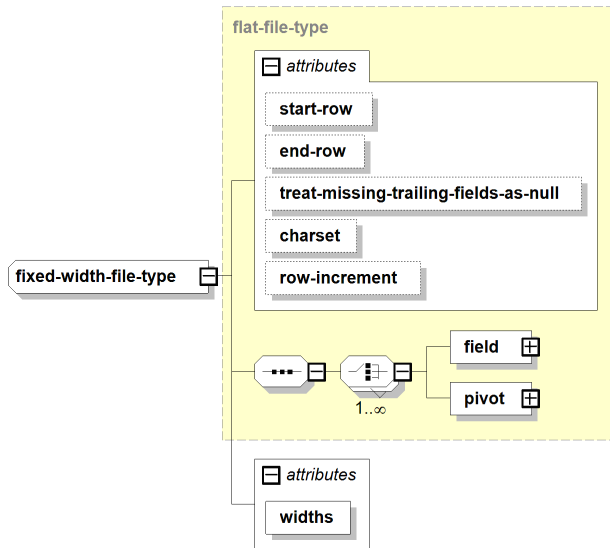
Required: For comma-delimited files, set `delimiter=" , "`. For tab-delimited files, set `delimiter="	 "` where `	` is the XML character denoting a tab. (You cannot use a tab character because tabs may be stripped away by XML parsers.)

text-qualifier

Characters between which to qualify text. Delimiters are treated as normal characters inside qualified text. The most common text qualifier is the double-quote; for example, `text-qualifier="""` where `"` is the XML character denoting a double-quote. (By default, text is not qualified.)

<fixed-width-file>

Fields in the data file are fixed in width.



start-row

Row from which to start loading. For example, `start-row="2"` tells the loader to skip the first row. (Default=1)

end-row

Row after which to stop loading. (Default=end of file or first empty row)

treat-missing-trailing-fields-as-null

Whether to treat missing trailing fields (files with fewer fields than defined in the load specification) as null values instead of errors. (Default=false)

charset

Character set used by the data file. Generally defined only for files that use a character set different from that of the computer on which the loader is running. (By default, the character set is not specified.)

row-increment

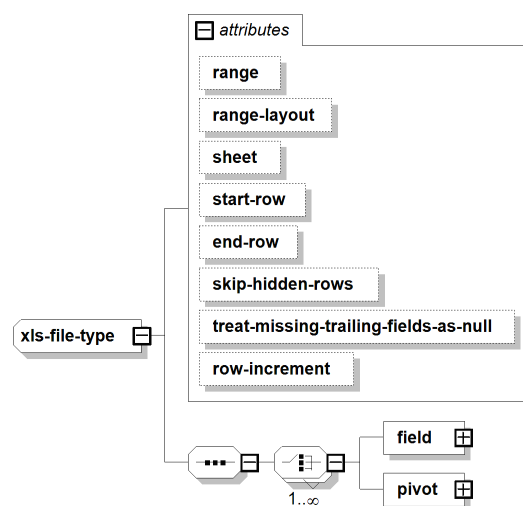
Number of rows between each record in the file. Generally defined only for files containing multiple data pivots. (Default=1)

widths

Required:. Widths of the fields in the data file. For example, `width="10 18 7"` means the first field is 10 characters wide, the second field is 18 characters wide, and the third field is 7 characters wide.

<ms-excel-file>

The data file is a Microsoft Excel workbook.



range

Excel named range from which to load. The named range must be confined to a single sheet. (Note that range and sheet are mutually exclusive.)

range-layout

Whether to arrange multiple segments horizontally (`range-layout="horizontal"`) or vertically (`range-layout="vertical"`). By default, the range layout is not specified.

sheet

Sheet from which to load. If neither sheet nor range is specified, the loader will load from the first sheet. The default is to load from the first sheet. (Note that sheet and range are mutually exclusive.)

start-row

Row from which to start loading. For example, `start-row="2"` tells the loader to skip the first row. (Default=1)

end-row

Row after which to stop loading. (Default=end of data or first empty row)

skip-hidden-rows

Whether to skip hidden rows. (Default=true)

treat-missing-trailing-fields-as-null

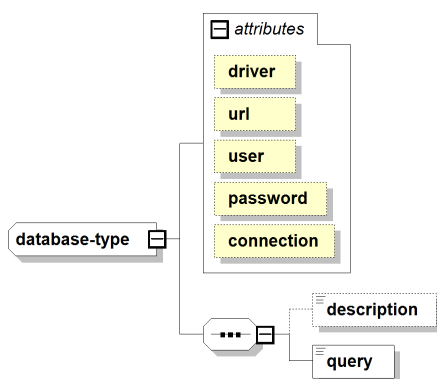
Whether to treat missing trailing fields (files with fewer fields than defined in the load specification) as null values instead of errors. (Default=false)

row-increment

Number of rows between each record in the file. Generally defined only for files containing multiple data pivots. (Default=1)

<database>

The data source is the result of a query against a database. The query is expressed in the <query> sub-element.



driver

Required: JDBC driver class name. For example:

- Oracle 9.2.0—`oracle.jdbc.driver.OracleDriver`
- SQL Server 2000—`com.microsoft.jdbc.sqlserver.SQLServerDriver`

The jar file containing the driver must be present in `<install-root>/lib`.

url

Required: URL to connect to the database. For example:

- Oracle 9.2.0—`jdbc:oracle:thin:@hostname:1521:dbname`
- SQL Server 2000—`jdbc:microsoft:sqlserver://hostname:1433;DatabaseName=dbname;SelectMethod=cursor`

where *hostname* is the name of the database server host, and *dbname* is the name of the database.

user

Required: User to log on to the database.

password

Required:. Password for the specified database user. Passwords beginning with "{3DES}" are assumed to be encrypted with the `<install-root>/bin/encrypt` utility.

connection

Name of the server's internal connection pool from which database connections are obtained. (Mutually exclusive with driver, url, user and password.)

Note: Starting with Release 2.5.1, you can omit *all* of the above attributes to signify that the database is the same one used by the Integrated Operational Planning Server.

<database> / <query>

Any SQL query that is valid for the database.

For example:

```
<database driver="oracle.jdbc.driver.OracleDriver"
  url="jdbc:oracle:thin:@host:1521:salesdb"
  user="joe"
  password="password"
  <query>
    select * from SalesOrders
  </query>
</database>
```

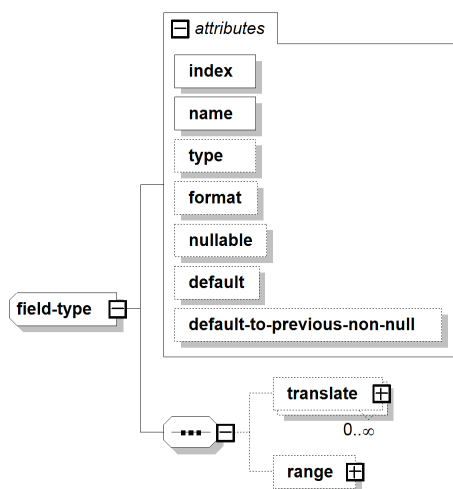
Query strings that contain XML meta-characters such as **>** must be marked as CDATA to prevent the XML parser from interpreting it. For example:

```
<query>
<![CDATA[
  select * from SalesPipeline where confidence > 0.9
]]>
</query>
```

The field names and types (used by the map elements) are taken from the column names and types of the query result set.

<flat-file> / <field>

Describes the fields in a flat (delimited or fixed-width) data file.



index

Required: Index position of the field. For example, the first field is `index="1"`.

name

Required: Name of the field. Give each field a unique name; for example, `name="part number"`.

type

Data type of the field. Valid types are: string, int, long, double, date, duration, and boolean. (Default=string)

format

Used for `type="date"` fields to tell the loader how to parse the date string. The format specification is as follows:

`yy`—two-digit year

`yyyy`—four-digit year

`MM`—two-digit month (01 to 12)

`MMM`—three-letter month (JAN, FEB, etc)

`dd`—two-digit day of month

For example, `format="MM/dd/yyyy"` equals 11/04/2004 and `format="yyyy-MM-dd"` equals 2004-11-04. Note that `yy` also matches four-digit years.

nullable

Whether the field can be null. If `nullable="false"`, the loader treats a null value as an error. (Default=true)

default

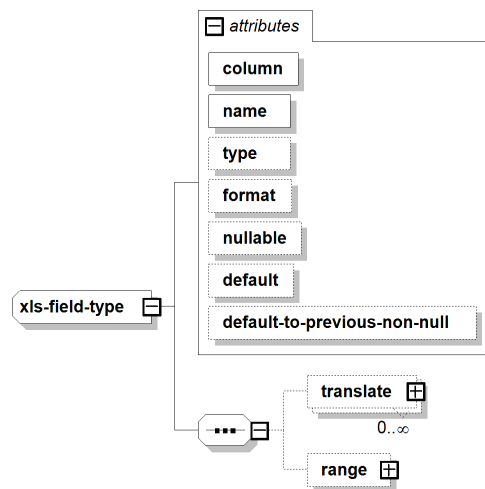
Value to substitute if the field value is null. (Default=null)

default-to-previous-non-null

Whether to use the previous non-null value if the current value of the field is null. The default applies if there is no previous non-null value. (Default=false)

<ms-excel-file> / <field>

Describes the fields in a flat (delimited or fixed-width) data file.



column

Required: Excel column identifier. For example, "A", "AB".

name

Required: Name of the field. Give each field a unique name; for example, name="part number".

type

Data type of the field. Valid types are: string, int, long, double, date, duration, and boolean. (Default=string)

format

Used for type="date" fields to tell the loader how to parse the date string. The format specification is as follows:

yy—two-digit year

yyyy—four-digit year

MM—two-digit month (01 to 12)

MMM—3three-letter month (JAN, FEB, etc)

dd—two-digit day of month

For example, `format="MM/dd/yyyy"` equals `11/04/2004` and `format="yyyy-MM-dd"` equals `2004-11-04`. Note that `yy` also matches four-digit years.

nullable

Whether the field can be null. If `nullable="false"`, the loader treats a null value as an error. (Default=`true`)

default

Value to substitute if the field value is null. (Default=`null`)

default-to-previous-non-null

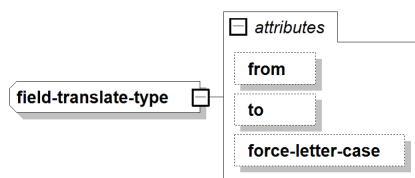
Whether to use the previous non-null value if the current value of the field is null. The default applies if there is no previous non-null value. (Default=`false`)

<field> / <translate>

A `<field>` can contain optional `<translate>` elements to convert the field value to another value. `<translate>` elements are typically used to force the field to all uppercase, or to replace aliases with a canonical value.

For example, the following two `<translate>` elements convert “Acme Corp.” and “Acme Corporation” to “Acme”:

```
<translate from="Acme Corp." to="Acme"/>
<translate from="Acme Corporation" to="Acme"/>
```



from

Fields that match the `from` value will be translated. (Default=`null`)

to

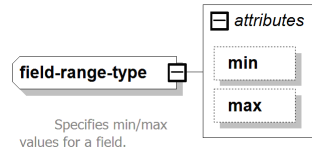
Fields that match the `from` value will be translated to this string. (Default=`null`)

force-letter-case

Translate the field to uppercase or lowercase. Case conversion occurs before translation. For example, `force-letter-case="uppercase"` converts the field value to all uppercase. (By default, there is no case conversion.)

<field> / <range>

<range> is an optional sub-element of <field> specifying that valid field values must be between the defined minimum and maximum. Field values that fall outside the range are flagged as errors. The data type and parse format are inherited from the enclosing <field> element. <range> is only applicable for date and numeric fields.



min

Minimum value for the field. If this is not defined, max must be defined. (Default=null)

max

Maximum value for the field. If this is not defined, min must be defined. (Default=null)

The following example specifies that valid build dates cannot exceed the end of 2100:

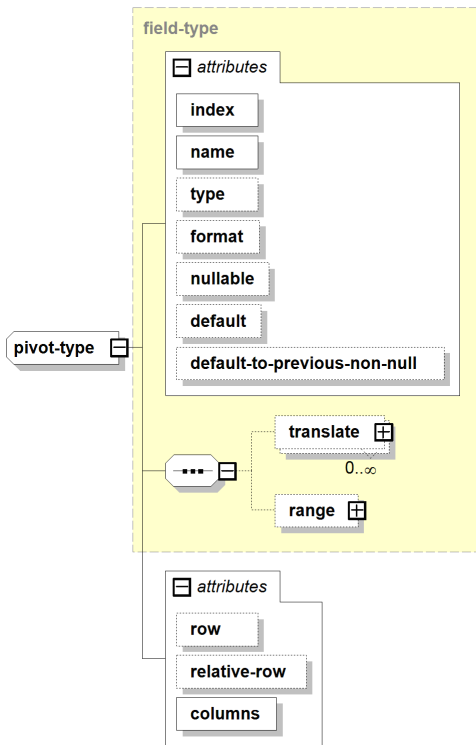
```
<field id="2" name="build date" type="date" format="MM/dd/yy">
  <range max="12/31/2100"/>
</field>
```

The following example specifies that the confidence value must be between 0.1 and 1.0:

```
<field id="8" name="confidence" type="double">
  <range min="0.1" max="1.0"/>
</field>
```

<flat-file> / <pivot>

A pivot is a special kind of array <field>; think of it as a row that is pivoted on the left and swung down to become a column. A file may contain more than one pivot row.



index

One-based column position of the pivoted array. No two <field>s or <pivot>s can have the same index. If omitted, the system assigns a unique index.

name

Required: Unique name of the pivot array. No two <field>s or <pivot>s can have the same name.

type

Data type of the elements in the pivot array. Valid values are: string, int, long, double, date, duration, and boolean. (Default=string)

format

Used with `type="date"` pivots to tell the loader how to parse the date string.

nullable

Whether pivot array elements can be null. If `nullable="false"`, the loader treats null values as errors. (Default=true)

default

Value to substitute if a pivot element is null. (Default=null)

default-to-previous-non-null

Whether to use the previous non-null value (from left to right before pivoting, or top-to-bottom after pivoting) if the current value of the pivot element is null. The default applies if there is no non-null previous value. (Default=false)

row

One-based row number that defines the pivot as a fixed pivot. The row attribute is not specified for data pivots, whose element values change from row to row. If you do not specify a row number, the pivot is defined as a data pivot.

relative-row

(For multiple data pivots only) The first instance of the data pivot row relative to the data file's start-row. For example, if the first data pivot starts on the same row as the start-row, then its `relative-row="0"`; if the second data pivot starts one row after the first, then its `relative-row="1"`. (Default=0)

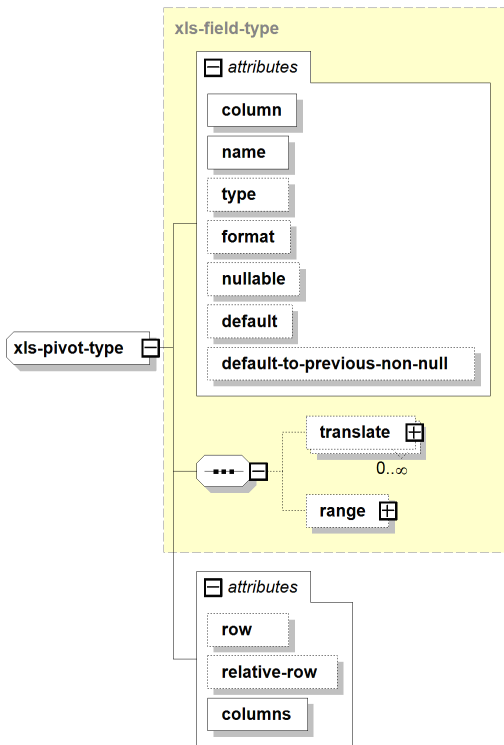
columns

Required: Columns in the pivot array specified as one or more comma-separated ranges: `col-a:col-b[,col-c:col-d]*`. The expression `col-a:col-b` means from column a to column b. Additional ranges can be specified in the same way, separated by commas. The intention is to omit columns (such as summarization columns) from the pivot.

For example: `2:6,8:12,14:18,20:24` defines a set of four column ranges starting at 2; each range has five columns and skips the column between the ranges. The columns may represent daily production numbers from Monday to Friday, and the skipped columns may represent weekly summarizations.

<excel-file> / <pivot>

The <pivot> element for Excel files is the same as for delimited or fixed-width files except that field positions are specified by *column* instead of by *index*.



column

Excel column identifier; for example, “A”, “AB”. If omitted, the system assigns a unique column identifier.

columns

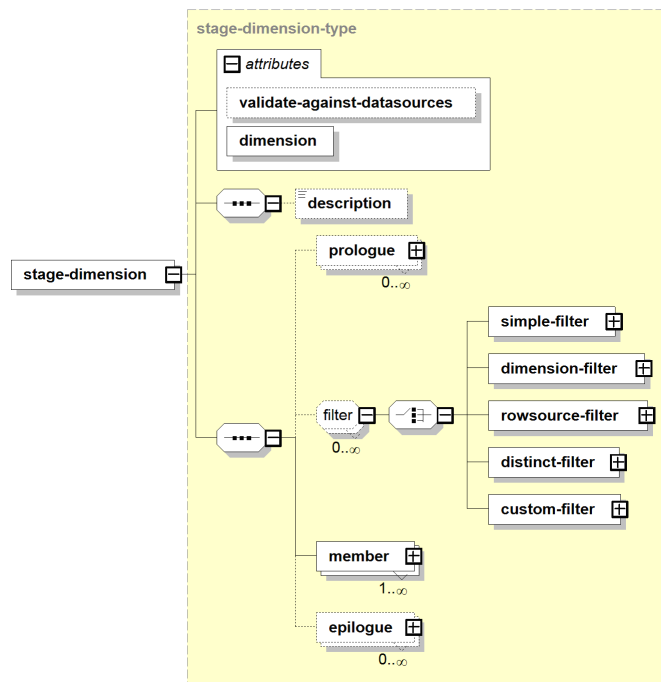
Required: Columns in the pivot array specified as one or more comma-separated ranges: `col-a:col-b[, col-c:col-d]*`. The expression `col-a:col-b` means from column a to column b. Additional ranges can be specified in the same way, separated by commas. The intention is to omit columns (such as summarization columns) from the pivot.

For example: `B:F, H:L` defines a set of two column ranges from B to F and from H to L, skipping column G. The columns may represent daily production numbers from Monday to Friday, and the skipped column may represent weekly summarizations.

All the other `<pivot>` attributes are the same as for delimited or fixed-width files.

<stage-dimension>

Fields from the data file are loaded into a dimension.



validate-against-data sources

Optional attribute applicable only for stagemaps. It has no effect for load specifications.

dimension

Required: Case-sensitive name of the dimension; for example, `dimension="Product"`. Define how the data fields are mapped to dimension members in the `<member>` elements. because Release 3.1, filters can be defined at this level to filter the load records before being processed by the `<member>` elements.

Following is an example of a specification to load the Product dimension:

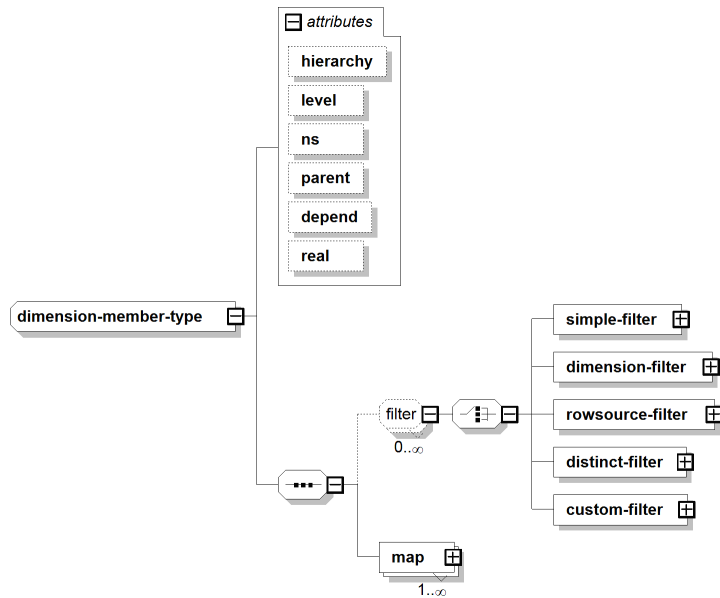
```
<delimited-file start-row="2" delimiter=",">
  <field index="1" name="product family"/>
  <field index="2" name="finished good"/>
</delimited-file>

<stage-dimension dimension="Product">
  <member hierarchy="Product">
    level="Product Family"
    ns="Product"
    parent="Product/Product/Product">
    <map attribute="name" value="{product family}"/>
  </member>
  <member hierarchy="Product">
    level="Model"
    ns="Product"
    parent="Product/Product/{product family}">
    <map attribute="name" value="{finished good}"/>
  </member>
</stage-dimension>
```

Note: `<prologue>` | `<epilogue>` enable for custom code to be invoked before and after staging.

`<stage-dimension>` / `<member>`

How the fields in the data file are translated to dimension members.



The following `<member>` attributes are all optional; the name of the member is specified in the `<map>` sub-element.

hierarchy

Name of the hierarchy to which the member belongs. If hierarchy is defined, level and parent must also be defined. (By default, the member does not belong to a hierarchy.)

level

Hierarchy level to which the member will be added. (By default, members are loaded just below the hierarchy root.)

ns

Namespace to which the member belongs. (By default, *ns* is the dimension's default namespace.)

parent

Fully-qualified name of the member's parent. The loader uses this information to add the member as a child of this parent in the hierarchy. (By default, the member is a child of the hierarchy root.)

depend

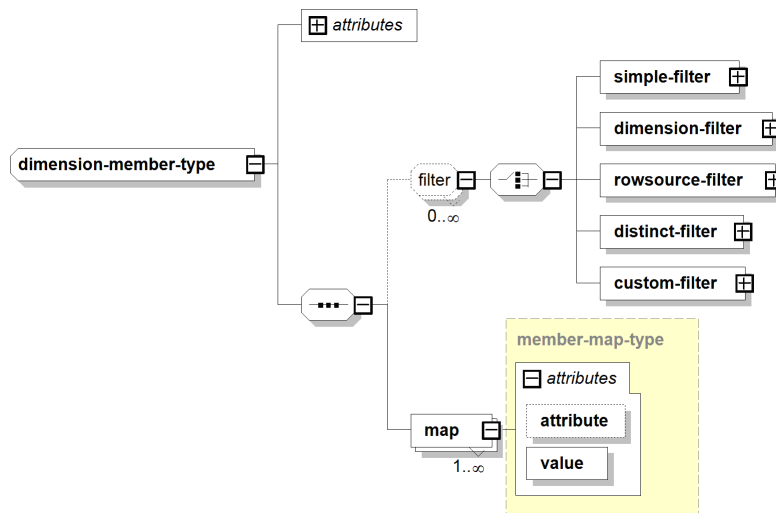
Overrides or bypasses certain levels in the hierarchy for calculation purposes. (By default, the depend value is the same as the parent.)

real

Defines that this member is an alias for another member whose fully-qualified name is given by the real value. (By default, the member is not an alias for another member.)

<stage-dimension> / <member> / <map>

<map> elements are used to specify the member's name, description, and displayName.



attribute

Name of the dimension member attribute. Valid attributes are name, description, displayName, and the name of any user-defined member attributes. (Default=name)

value

Required: Value of the member attribute.

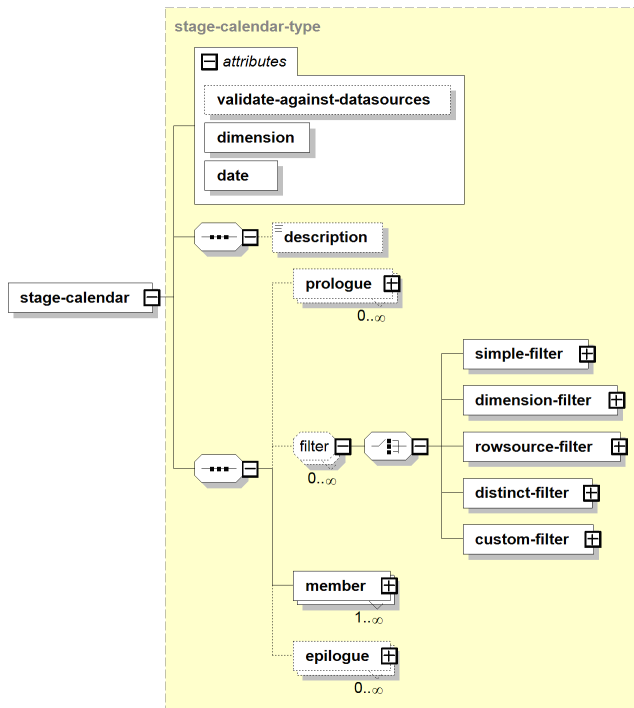
For example, the following <map> elements define the name and description for a member in the InventoryLocation dimension:

```
<member hierarchy="InventoryLocation"
  level="Location"
  ns="InventoryLocation" parent="InventoryLocation/InventoryLocation/
InventoryLocation">
  <map attribute="name" value="${location}" />
  <map attribute="description" value="${description}" />
</member>
```

Note: Data records can be filtered with only records matching the filter criteria let through to the <member> elements. Filter elements include <simple-filter>, <dimension-filter>, <rowsource-filter>, <distinct-filter>, and <custom-filter>.

<stage-calendar>

<stage-calendar> is a specialization of <stage-dimension> for loading time dimensions. It is the same as <stage-dimension> except for an additional date attribute.



validate-against-data sources

Optional attribute applicable only for stagemaps. It has no effect for load specifications.

dimension

Required: Case-sensitive name of the time dimension; for example, dimension="Fiscal".

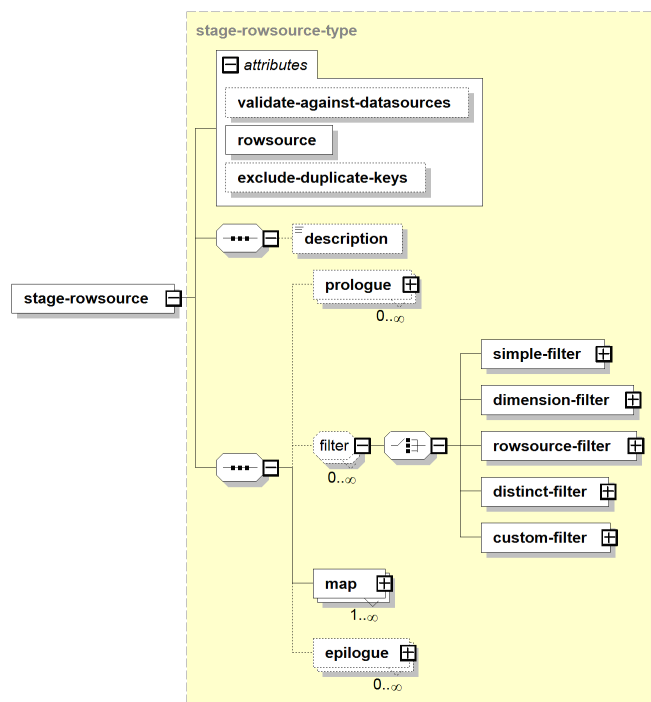
date

Required: Reference to the date field in the data file; for example, date="\${date}" defines the start of day, week, month, or quarter periods in the calendar.

Note: All sub-elements are the same as <stage-dimension>.

<stage-rowsource>

Defines the mapping of data file fields and rowsource columns.



validate-against-data sources

Optional attribute applicable only for stagemaps. It has no effect for load specifications.

rowsource

Required: Name of the rowsource.

exclude-duplicate-keys

Whether the loader should filter out load records whose key values are already staged. (Default=true) Note: Do not set to false for the parent rowsource of a parent-child pair.

Note: Data records can be filtered with only records matching the filter criteria let through to the <member> elements. Filter elements include <simple-filter>, <dimension-filter>, <rowsource-filter>, <distinct-filter>, and <custom-filter>.

Note: `<prologue>` | `<epilogue>` enable custom code to be invoked before and after staging.

<stage-rowsource> / <map>

`<map>` elements specify how to map data fields to rowsource column names and values.

column

Required: Name of the rowsource column.

value

Required: Value of the rowsource column. This can be a simple field reference such as `value="${location}"`, or an expression such as `value="${part}:${location}"`.

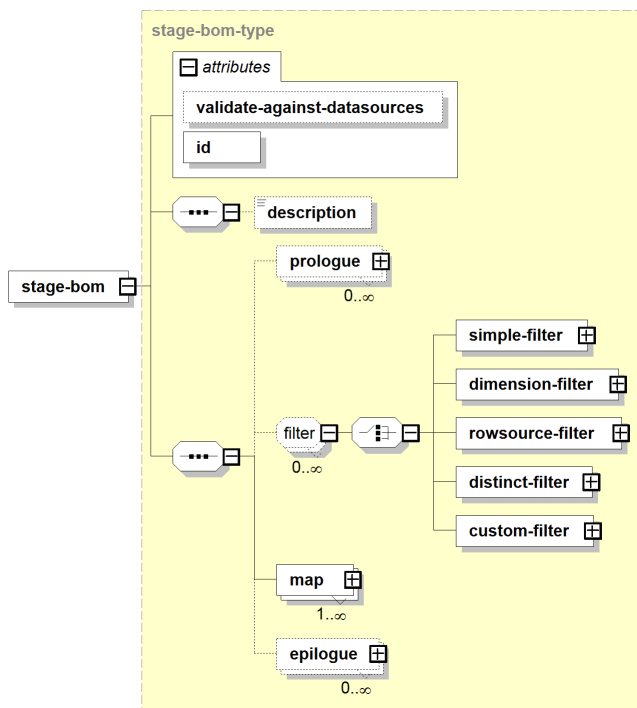
The following is a sample `<stage-rowsource>` specification:

```
<delimited-file start-row="2" delimiter=",">
  <field index="1" name="part"/>
  <field index="2" name="location"/>
  <field index="3" name="quantity" type="double"/>
  <field index="4" name="extraction date" type="date" format="MM/dd/yy"/>
</delimited-file>

<stage-rowsource rowsource="Inventory">
  <map column="part" value="${part}"/>
  <map column="location" value="${location}"/>
  <map column="quantity" value="${quantity}"/>
  <map column="inventoryDate" value="${extraction date}"/>
</stage-rowsource>
```

<stage-bom>

Maps fields in the data file to a bill of materials.



validate-against-data sources

An optional attribute applicable only for stagemaps. It has no effect for load specifications.

id

Required: Must be set to "BOM-0".

Note: Data records can be filtered with only records matching the filter criteria let through to the `<member>` elements. Filter elements include `<simple-filter>`, `<dimension-filter>`, `<rowsource-filter>`, `<distinct-filter>`, and `<custom-filter>`.

Note: `<prologue>` | `<epilogue>` enable custom code to be invoked before and after staging.

`<stage-bom>` / `<map>`

The `<map>` elements define the mapping between data fields and BOM entries.

attribute

Required: BOM entry attribute. Valid attributes include:

- **parent**—Assembly, or parent part
- **child**—Component, or child part
- **quantity**—Number of components used in the assembly
- **effective_date_begin**—Date from which the relation is valid
- **effective_date_end**—Date on which the relation is no longer valid

You must define `<map>` elements for parent, child and quantity.

value

Required: Value of the attribute.

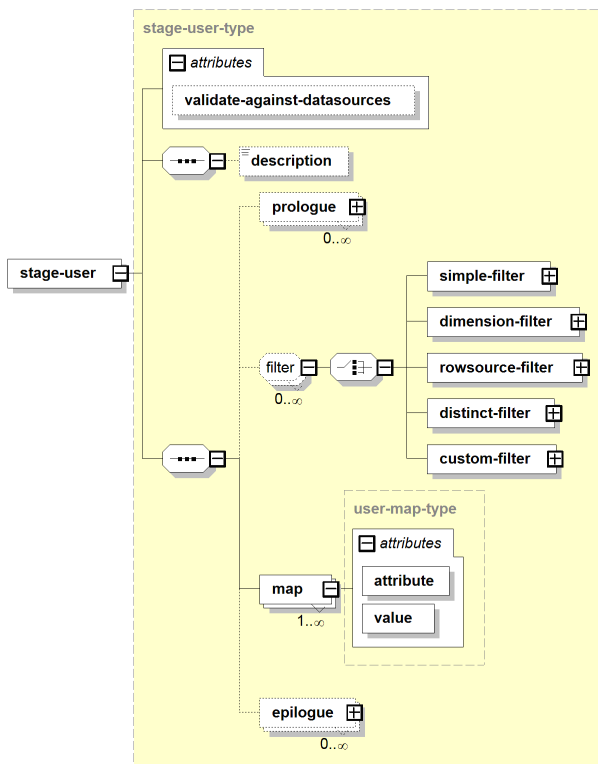
For example:

```
<delimited-file start-row="2" delimiter=",">
...<field index="1" name="assembly" nullable="false"/>
...<field index="2" name="component" nullable="false"/>
...<field index="3" name="quantity" nullable="false"/>
</delimited-file>
```

```
<stage-bom id="BOM-0">
  <map attribute="parent" value="{assembly}"/>
  <map attribute="child" value="{component}"/>
  <map attribute="quantity" value="{quantity}"/>
</stage-bom>
```

`<stage-user>`

Defines how fields in a file are mapped to user attributes.



validate-against-data sources

An optional attribute applicable only for stagemaps. It has no effect for load specifications.

<stage-user> / <map>

The `<map>` elements define a user's login name, password, descriptive name, and e-mail address.

attribute

Required: Valid attributes include:

- **login**—Login name
- **password**—Password
- **name**—Descriptive name for the user
- **email**—E-mail address

value

Required: Value of the attribute.

For example:

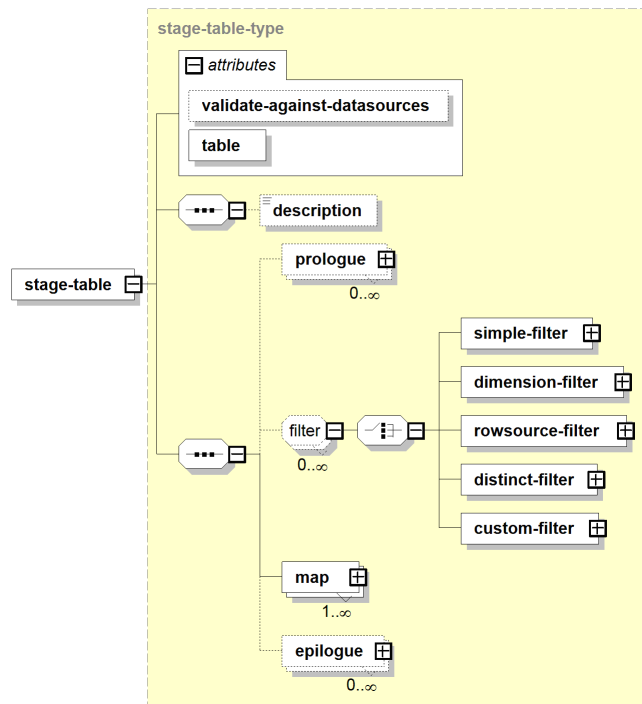
```
<delimited-file start-row="2" delimiter=",">
  <field index="1" name="login" nullable="false"/>
  <field index="2" name="password" nullable="false"/>
  <field index="3" name="name" nullable="false"/>
  <field index="4" name="email" default="demo@interlacesystems.com"/>
```

```
</delimited-file>
```

```
<stage-user>
  <map attribute="login" value="{login}" />
  <map attribute="name" value="{name}" />
  ...<map attribute="password" value="{password}" />
  ...<map attribute="email" value="{email}" />
</stage-user>
```

<stage-table>

Loads a simple database table.



table

Required: Database table name.

Note: `<stage-table>` is similar to `<stage-rowsource>`.

<stage-table> / <map>

The `<map>` elements define how to map data fields to table columns.

column

Required: Column name.

value

Required: Column value.

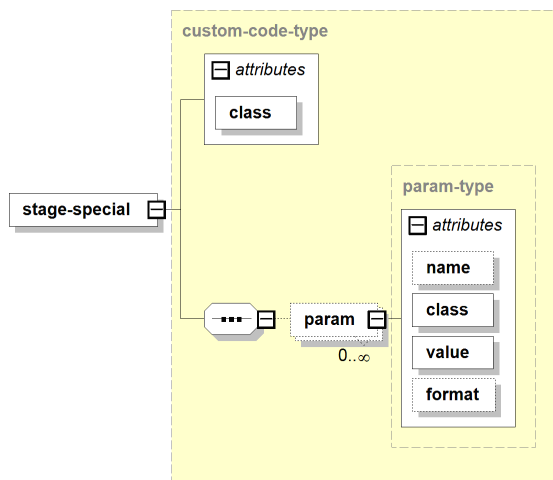
For example:

```
<delimited-file start-row="2" delimiter=",">
...<field index="1" name="id" nullable="false"/>
...<field index="2" name="channel"/>
...<field index="3" name="name"/>
</delimited-file>
```

```
<stage-table table="CUSTOMERS">
...<map column="id" value="{id}"/>
...<map column="class" value="{channel}"/>
...<map column="name" value="{name}"/>
</stage-table>
```

<stage-special>

Defines an invocation of a custom loader.



class

Required: Full Java class name of the custom loader; for example,

`class="com.acme.loader.MyCustomLoader"`

<stage-special> / <param>

`<param>` elements are used to define arguments to the custom loader's constructor.

name

Name of the parameter. This attribute is defined to document the argument. (By default, the name is not defined.)

class

Required: Full Java class name of the argument's type. For example, a string argument will set `class="java.lang.String"`.

value

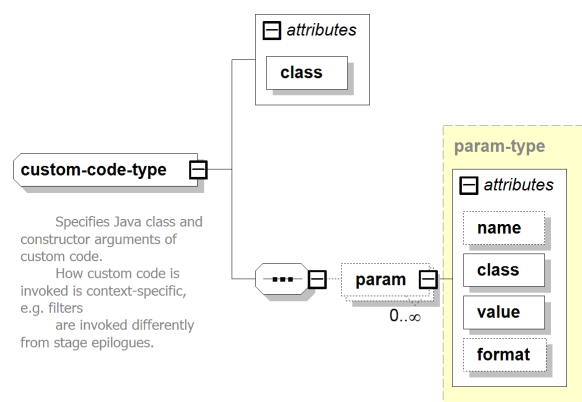
Required: Value of the argument.

format

Format of the value. This attribute is typically used for date arguments, and is used by the loader framework to parse the value string into the required class. (By default, the format is not defined.)

<prologue> | <epilogue>

Defines custom code to execute before and after staging.



For example:

```
<stage-rowsource rowsource="Parts">
  <map column="part" value="{part}"/>
  <map column="location" value="{location}"/>
  <map column="standardCost" value="{standard cost}"/>
  <epilogue class="com.acme.etl.CopyStandardCost">
    <param name="hq-location" class="java.lang.String" value="Xanadu"/>
  </epilogue>
</stage-rowsource>
```

The `<epilogue>` element in this example causes the following Java code to be executed on completion of the stage command:

```
StageEpilogue epilogue = new com.acme.etl.CopyStandardCost("Xanadu");
epilogue.epilogue();
```

class

Required: Name of the Java class that implements the custom `<prologue>` or `<epilogue>` code.

(<prologue> | <epilogue>) / <param>

Constructor parameters for custom Java classes are specified with <param> elements; these are omitted if the constructor does not take parameters.

name

Name of the parameter. This attribute is defined to document the argument. (By default, the name is not defined.)

class

Required: Full Java class name of the argument's type. For example, a string argument will set `class="java.lang.String"`.

value

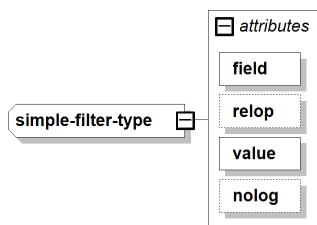
Required: Value of the parameter.

format

Format of the value. This attribute is typically used for date arguments, and is used by the loader framework to parse the value string into the required class.

<simple-filter>

Only records matching the filter criteria are let through to the <load> element.



field

Required: A reference to a field in the data file; for example, `field="{part}"`.

relop

One of the following values:

- EQ—equals (default value)
- NE—not equals
- GT—greater than
- GE—greater or equals
- LT—less than

- LE—less or equals
- IN—in
- NI—not in
- LIKE—regular expression match
- NOTLIKE—inverse of LIKE

value

Required: Value against which the field's value is compared. This can be a load-function expression if `relop` is EQ, NE, GT, GE, LT, or LE.

nolog

The filter criteria is expressed as a field `relop` value, and only those records for which the expression is true are let through.

Following are some examples using `<simple-filter>`:

Include only data records where `location` is not null:

```
<stage-rowsource rowsource="Parts">
  <simple-filter field="${location}" relop="NE" value="" />
  ...
</stage-rowsource>
```

Include only data records where `req_elem` is SB, AR or BB:

```
<stage-rowsource rowsource="MRP">
  <simple-filter field="${req_elem}" relop="IN" value="SB,AR,BB" />
  ...
</stage-rowsource>
```

Include only data records where `part` begins with `fg`:

```
<stage-rowsource rowsource="FinishedGoods">
  <simple-filter field="${part}" relop="LIKE" value="fg.*" />
  ...
</stage-rowsource>
```

Include only data records where `builddate` is after `exportdate`:

```
<stage-rowsource rowsource="PlannedBuilds">
  <simple-filter field="${builddate}" relop="GT" value="${exportdate}" />
  ...
</stage-rowsource>
```

Include only data records where `builddate` is in the future:

```
<stage-rowsource rowsource="PlannedBuilds">
  <simple-filter field="${builddate}" relop="GT" value="$today" />
  ...
</stage-rowsource>
```

Include only those records where `orderdate` is after 6/1/2009:

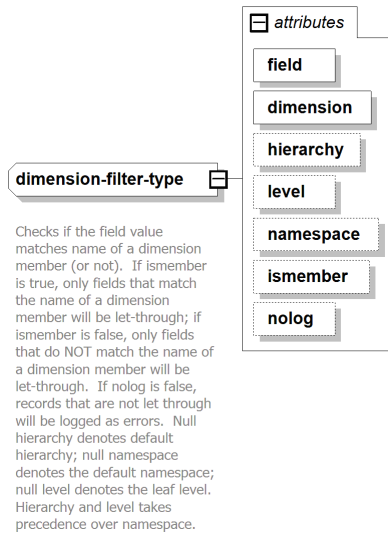
```

<stage-rowsource rowsource="Orders">
  <${orderdate}>
    relop="GE"
    value="${date(6/1/2009,MM/dd/yyyy)}"/>
</stage-rowsource>

```

<dimension-filter>

Compares field values against names of members in a dimension. When using <dimension-filter>, only non-shadow members of the filter dimension are matched.



field

Required: A reference to a field in the data file, such as `field="${part}"`, or an expression such as `field="${part}:${location}"`.

dimension

Required: Case-sensitive name of the dimension.

hierarchy

Limit the match to members in this hierarchy. The default hierarchy is assumed if hierarchy is not specified. Note that *namespace* takes precedence over *hierarchy* and *level*. (Default=default hierarchy)

level

Limit the match to members at this hierarchy level. The leaf level is assumed if hierarchy is specified, but level is omitted. Note that *namespace* takes precedence over *hierarchy* and *level*. (Default=leaf level)

namespace

Limit the match to members in this namespace. The default namespace is assumed if namespace is not specified. Note that *namespace* takes precedence over *hierarchy* and *level*. (Default=default namespace)

ismember

If *ismember*="true", only records whose field values match member names are let through. If *ismember*="false", only records whose field values do *not* match members names are let through. (Default=true)

nolog

If *nolog*="false", records that do not match are logged as errors. If *nolog*="true", non-matches are *not* logged as errors. (Default=false)

Following are some examples using <dimension-filter>:

Include only those data records where *part* matches the names of members in the *Part* dimension. Non-matching records are logged as errors.

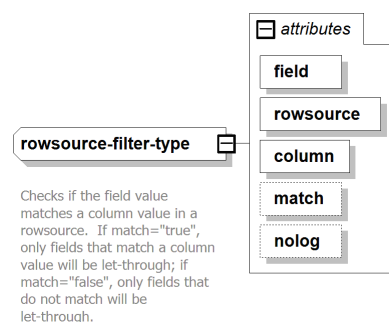
```
<stage-rowsource rowsource="Parts">
  <dimension-filter field="{part number}" dimension="Part"/>
  ...
</stage-rowsource>
```

Filter for finished goods by letting through only those records where *part* matches the names of members in the *Product* dimension at the *Finished Goods* level. Non-matching records are not logged as errors.

```
<stage-rowsource rowsource="FinishedGoodsInventory">
  <dimension-filter field="{part number}"
  <dimension-filter field="{part number}"
    nolog="true"/>
    level="Finished Goods"
  ...
</stage-rowsource>
```

<rowsource-filter>

Compares field values against column values in a rowsource.



field

Required: A reference to a field in the data file such as `field="{part}"`, or an expression such as `field="{part}:{location}"`.

rowsource

Required: Case-sensitive name of the rowsource.

column

Required: Case-sensitive name of the rowsource column against which the field values are matched.

match

If `match="true"`, only records whose field values match the column values are let through to the `<load>` element. If `match="false"`, only records whose field values do *not* match the column values are let through. (Default=true)

nolog

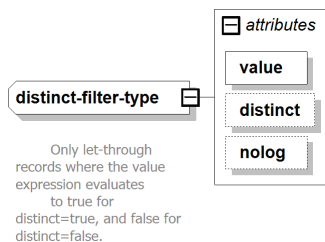
If `nolog="false"`, records that do not match are logged as errors. If `nolog="true"`, non-matches are *not* logged as errors. (Default=false)

The following example includes only those data records where the `part:location` value matches a value in the component column of the Parts rowsource. Non-matching records are logged as errors.

```
<stage-rowsource rowsource="CMDData">
  <rowsource-filter field="{part}:{location}"
    rowsource="Parts"
    column="component" />
  ...
</stage-rowsource>
```

<distinct-filter>

Lets records through based on whether an expression evaluated for each record is distinct.



value

Required: An expression that is evaluated for each record and whose value is used to determine whether the record is distinct. For example, `value="{part}"` means the value of the part field

is used to determine whether the record is distinct, and `value="{part}:{location}"` means the component string is used to determine whether the record is distinct.

distinct

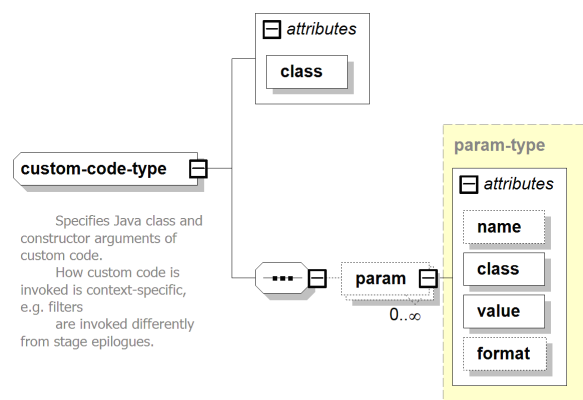
If `distinct="true"`, only records where the value expression is distinct are let through. If `distinct="false"`, only the second and subsequent records for which the value expression evaluates to the same value are let through. (Default=`true`)

nolog

If `nolog="false"`, records that do not match are logged as errors. If `nolog="true"`, non-matches are *not* logged as errors. (Default=`false`)

<custom-filter>

Defines code for custom filtering. The specification is the same as the other custom-code elements such as `<prologue>` | `<epilogue>` and `<stage-special>`.



Value Functions

The *value* attribute in the various `<map>` elements defines the value to assign to the dimension or rowsource attribute. Generally, it is a simple field reference, such as `"${location}"`. The value can also be an expression, such as `"$add(${date}, -P3D)"`.

The following functions can be used in a *value* expression:

`${name}`

`${name}` returns the value of the named field in the current data record. The `<field>` element is one whose *name* property matches `${name}`. If the field's *type* property is defined, the value will be an object of the appropriate type.

The following example returns the value of the location field:

```
${location}
```

\$add(lhs, rhs)

\$add(lhs, rhs) returns the sum of *lhs* and *rhs*. Typically used to add date and duration.

The following example subtracts three days from the value of `${date}`:

```
$add(${date}, -P3D)
```

\$begin

Subtopics

- `$begin(calendar)`
- `$begin(calendar, begin-level, date, offset-level, offset)`
- `$begin(calendar, level)`
- `$begin(calendar, level, date)`

\$begin(calendar)

\$begin(calendar) returns the start of the defined calendar

The following example returns 12/29/2009, the start of the Manufacturing calendar as defined in the calendar data file. This is useful for initializing the `effectiveTime` column when bootstrapping time-varying rowsources.

```
$begin(Manufacturing)
```

\$begin(calendar, begin-level, date, offset-level, offset)

\$begin(calendar, begin-level, date, offset-level, offset)—(1) Finds the member in the calendar at *begin-level* where *date* falls. (2) Finds the member at *offset-level* that starts on the same date as the member at *begin-level* (or failing that, the member at *offset-level* that contains the start date of the member at *begin-level*). (3) Finds the member at *offset-level* that is offset away from the member in (2) and returns its start date.

For example, if `$today` is 11/04/2006, the following expression will return the start of December in the Fiscal calendar. (The beginning of the current quarter is October, plus two months is December.)

```
$begin(Fiscal, Quarter, $today, Month, 2)
```

In the following example, `${month}` is of the form “M1”, “M2” for month 1, month 2.

`$substr(${month}, 1)` returns the month string from index position 1 after “M” (the numeric portion of the month string). `$add(..., -1)` converts the one-based month number to 0-based. `$begin()` returns the start of the current `${month}` in the Fiscal calendar.

```
$begin(Fiscal, Year, $today, Month, $add($substr(${month}, 1), -1))
```

`$begin(calendar, level)`

`$begin(calendar, level)` returns the start of the period in which today falls according to the defined calendar

The following example returns the start of the current week according to the Manufacturing calendar:

```
$begin(Manufacturing, Week)
```

This function is equivalent to `$begin(calendar, level, $today)`. The `level` parameter corresponds to a level in the default hierarchy of the calendar dimension. This function is a generalization of the `$week(calendar)`, `$month(calendar)`, and `$quarter(calendar)` functions.

`$begin(calendar, level, date)`

`$begin(calendar, level, date)` returns the start of the period in which the given date falls according to the defined calendar

The following example returns the start of the week in which `{build date}` falls according to the Manufacturing calendar:

```
$begin(Manufacturing, Week, {build date})
```

The `level` parameter corresponds to a level in the default hierarchy of the calendar dimension. This function is a generalization of the `$week(calendar, date)`, `$month(calendar, date)`, and `$quarter(calendar, date)` functions.

`$beginmember(qname, offset)`

`$beginmember(qname, offset)` returns the start of the period at the same level as the time-dimension member given by the fully-qualified name, and offsets number of periods away at the same level.

The following example returns the start of *Fiscal/FY 2005/November*:

```
$beginmember(Fiscal/FY 2005/August, 3)
```

For ForecastPro integration, the parameters can be field references; for example:

```
$beginmember({week}, {offset})
```

`$coalesce(arg1, ..., argN)`

`$coalesce(arg1, ..., argN)` returns the first non-null argument

The following example returns the value of `{territory}` if it is not null; otherwise, it returns the value of `{region}`:

```
$coalesce({territory}, {region})
```

\$date(string, format)

`$date(string,format)` returns a date object defined by the string, parsed according to the given format

For example:

```
$date(09/11/2001,MM/dd/yyyy)
```

The idea is to return a specific date to compare against other dates; for example:

```
<simple-filter field="{order date}"
              relop="GE"
              value="$date(6/1/2009,MM/dd/yyyy) "/"
>
```

where the *type* of `{order date}` is *date*.

\$datetime

Subtopics

- `$datetime(calendar, level)`
- `$datetime(calendar, level, date)`

\$datetime(calendar, level)

`$datetime(calendar,level)` returns the name of the time dimension members for the specified calendar at the specified level that correspond to the current server date

The following example returns the name of the current month according to the Fiscal calendar:

```
$datetime(Fiscal,Month)
```

\$datetime(calendar, level, date)

`$datetime(calendar,level,date)` returns the name of the time dimension member for the defined calendar at the defined level in which the given date falls

The following example returns the name of the week in the Manufacturing calendar in which `{build-date}` falls:

```
$datetime(Manufacturing,Week,{build-date})
```


\$quarter

Subtopics

- `$quarter`
- `$quarter(date)`
- `$quarter(calendar)`
- `$quarter(calendar,date)`

\$quarter

`$quarter` returns the start of the current quarter.

\$quarter(date)

`$quarter(date)` returns the start of the quarter in which the given date falls.

For example:

```
$quarter(${book date})
```

\$quarter(calendar)

`$quarter(calendar)` returns the start of the current quarter according to the given calendar.

For example, if today is January 5, 2005, and the current quarter in the Fiscal calendar is from November 28, 2004 to February 26, 2005, then the following expression will return 11/28/2004 (the beginning of the current quarter according to the Fiscal calendar):

```
$quarter(Fiscal)
```

\$quarter(calendar,date)

`$quarter(calendar,date)` returns the start of the quarter in which the given date falls according to the given calendar.

For example, if `${book date}` is January 5, 2005 and the quarter in the Fiscal calendar in which 1/5/2005 falls starts on 11/28/2004, then the following expression will return 11/28/2004:

```
$quarter(Fiscal,${book date})
```

\$quartername

Subtopics

- `$quartername(calendar)`
- `$quartername(calendar,date)`

\$quartername(calendar)

`$quartername(calendar)` returns the name of the current quarter as defined in the given calendar.

For example, if today is 6/5/2005, then the following expression will return “FY05 Q2” (the name of the current quarter as defined in the Fiscal calendar):

```
$quartername(Fiscal)
```

\$quartername(calendar,date)

`$quartername(calendar,date)` returns the name of the quarter in which the given date falls, as defined by the given calendar.

For example, if `${book date}` is 1/5/2005, the following expression will return “FY04 Q4” as the name of the quarter in which 1/5/2005 falls as defined by the Fiscal calendar:

```
$quartername(Fiscal,${book date})
```

\$lowercase(field)

`$lowercase(field)` returns the lowercase value of the argument.

For example, if the field value of `${name}` is “Fred”, the following expression will return “fred”:

```
$lowercase(${name})
```

\$map(field,test?a:b)

`$map(field,test?a:b)`—If the field matches test, returns a; otherwise returns b. A null test matches a null field value. If b is null, it means *field*.

For example, if the value of `${makebuy}` is “MAKE”, the following expression returns the value of `${make lead time}`; otherwise, it returns the value of `${buy lead time}`:

```
$map(${makebuy},MAKE?${make lead time}:${buy lead time})
```

If the value of `${territory}` is null, the following expression returns the value of `${region}`; otherwise, it returns the value of `${territory}`.

```
$map(${territory},?${region}:)
```

Note: The `$coalesce()` function is more efficient when looking for non-nulls.

\$mondaymonth

\$mondaymonth returns the date of the Monday on or before the current month.

\$month

Subtopics

- [\\$month](#)
- [\\$month\(calendar\)](#)
- [\\$month\(calendar,date\)](#)

\$month

\$month returns the start of the current month.

\$month(date) returns the start of the month in which the given date falls.

For example:

```
$month(${actual receipt date})
```

\$month(calendar)

\$month(calendar) returns the start of the current month according to the given calendar.

For example, if server date is 05-JAN-2005, and the current month in the Fiscal calendar is from 03-JAN-2005 to 06-FEB-2005, then the following expression will return 01/03/2005 (the beginning of the current month according to the Fiscal calendar):

```
$month(Fiscal)
```

\$month(calendar,date)

\$month(calendar,date) returns the start of the month in which the given date falls according to the given calendar.

For example, if `${book date}` is 05-JAN-2005, and the month in the Fiscal calendar in which 1/5/2005 falls starts on 03-JAN-2005, then the following expression will return 01/03/2005:

```
$month(Fiscal,${book date})
```

\$mul(lhs,rhs)

\$mul(lhs,rhs) returns the product of the *lhs* and *rhs* arguments. This is valid only if the data types of both *lhs* and *rhs* are numeric (int, long or double).

The following example returns the product of `${confidence level}` and 100:

```
$mul(100,${confidence level})
```

\$parent (dimension, hierarchy, namespace, member)

`$parent(dimension,hierarchy,namespace,member)` returns the parent of the given member in the hierarchy belonging to the dimension. The *hierarchy* and *namespace* arguments are optional and only need to be defined if they are a different value than the default hierarchy and namespace.

The following example returns the parent member name of the member whose name is the value of the `${finished good}` field, in the Product dimension.

```
$parent(Product,,${finished good})
```

\$rowsource (rowsource, column, key1, expr1, ..., keyN, exprN)

`$rowsource(rowsource,column,key1,expr1,...,keyN,exprN)` returns the value of the column in the rowsource for the defined rowsource key. The rowsource key is defined as key-expr pairs.

The following example returns the customer group whose ID is given by the value of `${customer id}`. In this example, the Customers rowsource has only one key column, ID.

```
$rowsource(Customers,group,id,${customer id})
```

The following example returns the standard cost of the part whose part number and location is given by the values of `${part}` and `${location}`. In this example, part and location are the key columns for the Parts rowsource.

```
$rowsource(Parts,stdCost,part,${part},location,${location})
```

\$rsubstr (string, rbegin, rend)

`$rsubstr(string,rbegin,rend)` returns the substring of *string* starting at *rbegin* from the end of *string* and ending before *rend* from the end of *string*. If *rend* is omitted, it is omitted to the end of *string*. Both *rbegin* and *rend* are 0-based relative to the end of the *string*, from right to left. This function extracts the last *n* characters of a string.

The following example returns the last four characters of `${sku}`. If the value of `${sku}` is “FPB-6473”, it will return “6473”. `$rsubstr()` is useful for extracting the last *n* characters of a variable length field; for example, if the value of `${sku}` is “ABCD-1234”, it will return “1234”.

```
$rsubstr(${sku},3)
```

The following example returns the value of `${part number}` starting with the 7th character from the right, up to but excluding the 4th character from the right. If the value of `${part number}` is “1234-567-001”, then it will return “567”.

```
$substr(${part number},6,3)
```

This is similar to `$substr()` except that the indices are relative to the end of string, from right to left.

\$seq

Subtopics

- [\\$seq](#)
- [\\$seq\(name\)](#)

\$seq

`$seq` returns the next value in the global unnamed sequence. The sequence is not persistent.

\$seq(name)

`$seq(name)` returns the next value in the named sequence. The sequence is not persistent.

The following example returns the next value in the sequence whose name is the value of `${po number}`:

```
$seq(${po number})
```

\$subfield(field,delimiterChar,index)

`$subfield(field,delimiterChar,index)` returns the element at the defined 0-based index position of an internally-delimited field. If the data source is delimited, the delimiter character must either be different from the delimiter used in the field; otherwise, the field is text-qualified.

For example, if `${desc}` is “FPB-6473:East:Fiscal/FY 2005/May”, then the following expression will return “Fiscal/FY 2005/May”.

```
$subfield(${desc},:,2)
```

\$substr(s,begin,end)

`$substr(s,begin,end)` returns the substring of *s* starting at 0-based index positions *begin* and *end*. If *end* is omitted, it means to the end of *s*.

The following example returns the first three characters of `${finished good}`. If the value of `${finished good}` is “FPB-6473”, this will return “FPB”.

```
$substr(${finished good},0,3)
```

The following example returns the value of `${part number}` starting with the fifth character, and omitting the first four characters:

```
$substr(${part number}, 4)
```

See also `$rsubstr()`.

\$today

\$today returns today's date according to the server

\$trim(expr)

\$trim(expr) removes leading and trailing spaces from the value of the argument, which must be a field reference or an expression that evaluates to a string

For example, if \${name} is "Fred ", then the following expression will return "Fred":

```
$trim(${name})
```

In the following example, if the value of \${partno}:\${location} is " P1234-56-C:SJC ", the expression will return "P1234-56-C:SJC":

```
$trim(${partno}:${location})
```

\$uppercase(expr)

\$uppercase(expr) returns the uppercase value of the argument, which must be a field reference or an expression that evaluates to a string

For example, if the value of \${name} is "Fred", then the following expression will return "FRED":

```
$uppercase(${name})
```

If the value of the \${partno}:\${location} is "p1234-56-c:SJC", the following expression will return "P1234-56-C:SJC":

```
$uppercase(${partno}:${location})
```

\$week

Subtopics

- [\\$week](#)
- [\\$week\(calendar\)](#)
- [\\$week\(calendar,date\)](#)
- [\\$week\(date\)](#)

\$week

\$week returns the start of the current week. According to standard conventions, Sunday is the first day of the week.

\$week(calendar)

\$week(calendar) returns the start of the current week according to the given calendar

For example, if the server date is 05-JAN-2005, and the current week in the Manufacturing calendar is from 03-JAN-2005 to 09-JAN-2005, then the following expression will return 01/03/2005, the beginning of the current week according to the Manufacturing calendar:

```
$week(Manufacturing)
```

\$week(calendar,date)

\$week(calendar,date) returns the start of the week in which the given date falls according to the given calendar

For example, if `${build date}` is 05-JAN-2005, and the month in the Manufacturing calendar in which 1/5/2005 falls starts on 03-JAN-2005, then the following expression will return 01/03/2005:

```
$week(Manufacturing,${build date})
```

\$week(date)

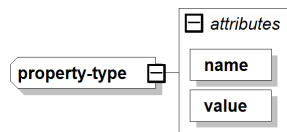
\$week(date) returns the start of the week in which the given date falls

For example:

```
$week(${actual build date})
```

Properties

Properties are expressed as `<property>` elements.



name

Name of the property.

value

Value of the property.

batch.size

Number of records batched together per database write. Large batch sizes do not necessarily improve performance. While batching records can have a significant impact with Oracle 9.2 (5ive times faster or more in some cases), the performance gain with SQL Server 2000 is not significant. (Default=1)



Block Cache Estimation

In This Appendix

Parameters	281
Equation	281
Example	281
Dense Dimensions and Java Heap Size	282

This appendix shows how to estimate the size of block cache in Integrated Operational Planning.

Parameters

- **Z**—Total memory size in GB
- **m**—Total number of measures in a block
- **t**—Total number of time members in a block
- **c**—Block cache size (the one you are going to estimate)
- **r**—Ratio of size occupied by block cache to the total memory

Equation

The equation relating the block cache to the total memory size is:

$$r = (c * m * t * 8) / (Z * 10^9)$$

This results in the following equation for block cache size estimation:

$$c = (r * Z * 10^9) / (m * t * 8)$$

Example

Assume you have the following:

- **Z**—8 GB
- **m**—100
- **t**—4 years

Each year has 69 members (52 weeks, 12 months, 4 quarters, and 1 year); so, 4 years has a total of 276 members. Round this to 300.

- $r=0.5$

You want only the blocks to occupy 50 percent of the total memory.

Plugging these numbers into the block cache size estimation equation:

$$c = (0.5 * 8 * 10^9) / (100 * 300 * 8)$$

results in a block cache size of 16667.

Dense Dimensions and Java Heap Size

The number of dense dimensions members affects the size of a block which ultimately affects the java heap size. For example, if the block cache is set to 2000, then the heap requirement for the blocks is 2000 times the size of the single block.

This example has the following three dense dimensions:

- Time dimension = 104 members (2 years at 52 weeks per year)
- Measure dimension = 30 members
- Version dimension = 10 members

In this case, the size of each block is $104 \times 30 \times 10 \times 8$ (bytes) = 249.6KB, and the total heap required is 499.2MB ($2000 \times 249.6\text{KB}$).

With the exception of time and measure dimensions, do not create dimensions as dense as the one in the above example.



Troubleshooting

In This Appendix

Configuration Issues	283
System and Environment Issues.....	284
Connection Issues	285
Deleting Objects.....	285
Sizing.....	285
Load Commands	286
Excel-Related Issues.....	286

Configuration Issues

Subtopics

- [Properties Files \(from least to highest\)](#)
- [Defaults](#)
- [Customer Specific](#)

Properties Files (from least to highest)

Configure the following properties in the following order:

```
INSTALL_ROOT/interlace/config/ISServer.properties
INSTALL_ROOT/interlace/config/application.properties
INSTALL_ROOT/custom/config/site.properties
INSTALL_ROOT/custom/config/MACHINE_NAME.properties (top priority)
```

Defaults

The following section lists the properties and the file locations where the default values are assigned:

- **host, port:** ISServer.properties/site.properties
- **DB:** ISServer.properties/site.properties
- **memory settings:** ISServer.properties
- **cache sizes:** site.properties

- **mail settings:** `IServer.properties`

Customer Specific

Place implementation specific properties in `MACHINE_NAME.properties` for each instance; for example: for development, test, and production instances.

IServer, application and site properties are common to all three instances. Therefore, configure the following in `MACHINE_NAME.properties`, which might be different for development, test and production instances: DB, memory, host, and port number.

System and Environment Issues

Subtopics

- [Out of Memory Error](#)
- [Database Issues](#)
- [E-mail Issues](#)
- [The Server is Not Listening on the Port](#)

Out of Memory Error

For 32-bit systems, 1024 MB is the recommended maximum allowable memory allocated to Java for running WebLogic.

For 64-bit systems, a recommended range of 4-6 GB depending upon system resources.

Set the default memory allocation properties in:

`INSTALL_ROOT/interlace/config/IServer.properties.`

Set the instance specific memory settings in:

`INSTALL_ROOT/custom/config/MACHINE_NAME.properties`

If possible, increase memory size allocation. Scripts generally utilize the largest amount of system resources.

To reflect changes, you must refresh the server.

Database Issues

If the database is down, check the following file for SQL exceptions and socket errors to identify any database connectivity issues:

`EPM_HOME/user_projects/domains/EPMSystem/servers/IOPServer_iopinstance1/logs/IOP.log`

If exceptions or errors are identified, note the time stamp and review them with the database administrator. The database administrator must check for any issues with the database server.

E-mail Issues

For e-mail issues, try the following:

- Verify the e-mail configuration settings in `ISServer.properties` and other properties files. See [“Properties Files \(from least to highest\)” on page 283](#).
- Ping the mail host and see if you receive a response.

The Server is Not Listening on the Port

The Integrated Operational Planning server port is configured in the properties file. When Integrated Operational Planning uses port 27080, it may conflict with other applications. If other applications use the same port as Integrated Operational Planning, specify another port.

Connection Issues

If you use Oracle Internet Directory (OID) users and an error message appears when you run a query link, verify that the Oracle's Hyperion® Shared Services server is started.

Deleting Objects

When you delete objects, you must first delete object mappings. Available mappings include cube-to-cube, row source-to-cube, data source-to-row source, data source-to-dimension, and allocation maps. See [“Creating Maps” on page 99](#).

- To delete the mappings, in the Model, go to the **Object Details Graph** tab, click the line between the objects, and select **Delete Mapping**.

Sizing

You can use a Microsoft Excel spreadsheet to calculate the total JVM memory needed.

- To calculate the amount of JVM Memory that you need:
 - 1 Open `INSTALL\samples\sample\jvm_sizing.xls`.
 - 2 Enter the dense and sparse dimension information into the dense and sparse dimension fields.
 - 3 In **Percentage of blocks that exist**, run a select distinct query on the distinct combinations of sparse dimension members that exist in the historical tables.
 - 4 In **Block-Cache Memory Percentage**, enter the percentage of the actual number of blocks in disk to be in the block-cache. As the percentage increases, the performance is enhanced, but you must then increase the memory.
 - 5 The Excel spreadsheet calculates the required total JVM memory.

Load Commands

If data does not get loaded into a cube measure:

1. Check *install-root/errors* for any errors involving the row source that loads data into the measure.

Errors types:

- A row source value does not match any member name.
Implies that there is data for some members that is not in a dimension. You can ignore these errors.
- A row source has duplicate values.
Implies that the key for row source is not properly defined. Look at the key and the mapping between the row source and the cube.

2. If there are no errors, look for the statement in the log that does the row source staging. Statements appear similar to: `stage replace using stagemap stagemap-name` or `stage update using stagemap stagemap-name`.
 - If there is no stage statement, the staging was never done. This usually happens if the script misses the stage command, or if the group name in the stagemap does not match the group name in the macro stage command.
 - If the stage statement exists, look for how many records were accepted or rejected. If there are no rejected records, then there is no problem with staging.
3. Check if the `load` command is in the log. It has the pattern “load update” or “load replace” into the row source that is mapped to the measure.
 - If there is no load statement, fix the scripts to add the statement.
 - If the load statement exists, observe how many records were accepted or rejected. Rejected records appear in the error file described previously.

Excel-Related Issues

Subtopics

- [Installing the Anteros Ambassador Active-X Control for Excel](#)
- [Launching Excel When You Are Not Connected to the Internet](#)
- [Loading From an Excel File](#)
- [Troubleshooting Excel's Failure to Launch from the Planning Workbench](#)

Installing the Anteros Ambassador Active-X Control for Excel

The Anteros Ambassador Active-X Excel plug-in can only be installed while connected to the Integrated Operational Planning server. Although the Ambassador is third-party software, Oracle does not have an install package or executable for the plug-in. Therefore, each new Integrated Operational Planning client PC that is deployed must complete the following steps:

1. In order to install and run the Active-X control for Excel, connect to the Integrated Operational Planning server while logged on to the computer using a Windows or domain account with local administrative or power user privileges.
2. Perform an analysis on a new or existing scenario to load the Ambassador plug-in.
See [“Performing an Analysis” on page 52.](#)
3. Close the analysis session, log out of the PC, and then log back in as a member of the local group.
4. Connect to the Integrated Operational Planning server and perform the analysis as usual.

After you complete these steps, the Ambassador plug-in for Excel will launch.

Launching Excel When You Are Not Connected to the Internet

Launching Excel to perform an analysis can take a long time if you are not connected to the Internet. To avoid this, ensure that the following registry entry contains the value CODEBASE:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet  
Settings\CodeBaseSearchPath
```

Loading From an Excel File

The path to automatically generate a loadspec from an Excel file and to automatically generate a table definition to support the load is as follows:

Excel, then load-spec, table generation, then drop/create those tables, then load excel to table, and then stage/load from table to rowsource.

You can re-use the loadspec for uploadable reports.

See [“Auto Generate Loader Specification” on page 163.](#)

Troubleshooting Excel's Failure to Launch from the Planning Workbench

If Excel does not launch from the Planning Workbench in Integrated Operational Planning, try the following:

- In Excel, select **Tools**, then **Macros**, and then **Security**. Switch to **Trusted Publishers** and ensure that **Trust all installed add-ins and templates** is selected.
- In Excel 2009 only, Select **Help**, and then **About Microsoft Excel**. Select **Disabled Items for Ambassador**. If listed, re-enable it again.
- Check for the following registry entry:

```
HKEY_CLASSES_ROOT\Excel.Application
```

If it is missing, copy the version from:

HKEY_CLASSES_ROOT\Excel.Application.9

- Uninstall Ambassador. (You may have to close all Microsoft Office applications including Outlook and Internet Explorer.) In Internet Explorer, select **Tools**, and then **Internet Options**; then, delete cookies and files.
- Complete the following steps:
 1. Uninstall Ambassador. (You may have to close all Microsoft Office applications including Outlook and Internet Explorer.)
 2. In Internet Explorer, select **Tools**, and then **Internet Options**; then, delete cookies and files.
 3. Remove the cache folder under Documents and Settings\your_login\Local Settings\Temp\interlace.
 4. Verify that your registry entry, HKEY_CURRENT_USER\Software, does not list an interlace entry. If the entry for interlace exists, remove it.
 5. In the registry, search for and remove the following keywords (in the Find dialog box, make sure to select *keys*, *values*, and *data*):
 - anteros
 - ambassador
 - {Anteros CLSID} - current: 79B589EC-406A-45a8-88C8-C07A3C7101DB
 6. Reboot your computer.
- Verify the following registry entries as used by Ambassador:
 - The following keys are used for creating important objects. If any are missing, the result is usually fatal.

HKEY_CLASSES_ROOT\CLSID{2933BF90-7B36-11d2-B20E-00C04F983E60} //XML DOM object
HKEY_CLASSES_ROOT\CLSID{ED8C108E-4349-11D2-91A4-00C04F7969E8} //XML HTTP Request
HKEY_CLASSES_ROOT\CLSID{8856F961-340A-11D0-A96B-00C04FD705A2} //the MS WebBrowser control
HKEY_CLASSES_ROOT\CLSID{0E59F1D5-1FBE-11D0-8FF2-00A0D10038BC} //the MS ScriptControl

- Ambassador reads configuration information from the following paths:

HKEY_CLASSES_ROOT\InternetExplorer.Application\
HKEY_CLASSES_ROOT\Excel.Application\
HKEY_CLASSES_ROOT\MSProject.Application\
HKEY_CLASSES_ROOT\Outlook.Application\
HKEY_CLASSES_ROOT\PowerPoint.Application\
HKEY_CLASSES_ROOT\Visio.Application\
HKEY_CLASSES_ROOT\Word.Application\
HKEY_CLASSES_ROOT\Word.Document\
HKEY_CLASSES_ROOT\MSXML2.DOMDocument\
HKEY_CLASSES_ROOT\Msxml2.DOMDocument\
HKEY_CLASSES_ROOT\MSXML2.DOMDocument.2.6\
HKEY_CLASSES_ROOT\Msxml2.DOMDocument.3.0\
HKEY_CLASSES_ROOT\MSXML2.DOMDocument.4.0\

- Collection of paths that Ambassador uses. Ambassador reads and writes from the following keys, mostly during installation:

```
HKEY_CLASSES_ROOT\.xml
HKEY_CLASSES_ROOT\xmlfile
HKEY_CLASSES_ROOT\CLSID\interlace.*
HKEY_CURRENT_USER\Software\interlace
HKEY_LOCAL_MACHINE\Software\Microsoft\Code Store Database\Distribution Units\
HKEY_LOCAL_MACHINE\Software\Microsoft\Office\Excel\Addins\
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Management\
\ARPCache\
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\ModuleUsage\
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SharedDLLs\
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\
```

Additionally, Ambassador maps from file extension to application using the registry by looking up the extension (for example, .xls) in the registry. The value for this key is a *progID*. If you look up the *progID* under HKEY_CLASSES_ROOT, you will find information on starting the appropriate application. Ambassador uses this information both directly and indirectly. You may encounter problems if you use a computer with incorrect Excel configurations.

- Check the following registry:

```
HKEY_CLASSES_ROOT\.xls
```

for the following setting:

```
HKEY_CLASSES_ROOT\.xls @="ExcelViewer.Sheet.8" "Content
Type"="application/vnd.ms-excel"
```

The user's registry has an application called “ExcelViewer” listed as responsible for .xls files. It should appear as follows:

```
HKEY_CLASSES_ROOT\.xls @="Excel.Sheet.8" "Content
Type"="application/vnd.ms-excel"
```

This is an indication that at some time the user (or an administrator) installed the Excel Viewer application over Excel. Changing the registry (as indicated above) will fix the problem. Reinstalling Excel should also correct the problem.

Glossary

administrator Analysis role in Integrated Operational Planning with a typical workflow of using the Administration Workbench to perform administrative tasks.

analysis owner Analysis role in Integrated Operational Planning with a typical workflow of creating scenarios to perform an analysis.

approver Analysis role in Integrated Operational Planning with a typical workflow of approving proposed plan changes developed as the result of an analysis.

assigner Analysis role in Integrated Operational Planning with a typical workflow of monitoring exceptions in the base data and assigning new exceptions for users to investigate on their home page.

constraint A configured business rule or threshold.

cube A block of data that contains three or more dimensions.

cube definition An XML file that defines the dimensions in a cube, the names of measures in the cube and measure formulas, the names of row sources mapped to the cube, and cube-to-cube mapping information.

data provider Analysis role in Integrated Operational Planning with a typical workflow of receiving e-mailed report workbooks and providing updated report data.

data source A relational table in the Integrated Operational Planning database where copies of external modeling data are stored.

data source definition An XML file that defines what data source columns to use.

dense dimension In block storage databases, a dimension likely to contain data for every combination of dimension members.

dimension A data category used to organize business data for retrieval and preservation of values. Dimensions usually contain hierarchies of related members grouped within them.

dimension definition An XML file that defines the namespace name, hierarchy names, and hierarchy level names for a dimension.

exceptions Values that satisfy predefined conditions. You can define formatting indicators or notify subscribing users when exceptions are generated.

execution database Original source of data to load in to Integrated Operational Planning.

Extensible Markup Language A language comprising a set of tags used to assign attributes to data that can be interpreted between applications according to a schema.

hierarchy The organization of multidimensional dimension members in an outline.

Integrated Operational Planning database Data in Integrated Operational Planning used in what-if scenarios.

ISA file A file that contains a list of isadmin commands. To execute this file, enter `isadmin -u admin -p password -f <isa file with full path>`.

isadmin A client-side command-line tool that interacts with the Integrated Operational Planning server.

key assumption A driver behind the underlying planning model. Similar to a key metric in the sense that it represents intersections in a cube; however, a key assumption represents a driver behind the planning model while a key metric represent the results from the planning model. Examples of key assumptions include future prices, inflation, currency exchange rates, and labor rates.

key metric One or more intersections in a cube that are of significant business interest. .

loader Component that stages and loads to data sources, row sources, and cube cells.

MDX (multidimensional expression) The language that gives instructions to OLE DB for OLAP- compliant databases, as SQL is used for relational databases.

measure A dimension whose members are the measures calculated in an analysis.

namespace A mapping that matches dimension member names to internal system IDs.

participant Analysis role in Integrated Operational Planning with a typical workflow of collaborating on an analysis to review scenarios and providing comments to the analysis owner.

planning workbook A predefined Excel workbook used in what-if analysis.

planning worksheet A subset of modeling data in the Integrated Operational Planning database.

reconcile A process by which open scenarios resolve their changes with new changes submitted to the baseline. A new baseline is started and the changes for each scenario are reapplied.

report worksheet A predefined Excel worksheet that uses tabular reports and charts to display data in the Integrated Operational Planning database.

row source A relational table in the Integrated Operational Planning database that contains data values.

row source definition An XML file that defines the columns in a row source and the parent-child relationships between row sources.

RSQL (Row Source Structured Query Language) Uses SQL to return row source data. Row source data is transactional, component-level data loaded into relational tables in the Integrated Operational Planning database from an execution database.

sandbox Internal and external data structures that support the ability to do changes in isolation for different users.

scripts Commands invoked to perform specific activities in Integrated Operational Planning.

sparse dimension In block storage databases, a dimension unlikely to contain data for all member combinations when compared to other dimensions.

stagemap An XML file that maps data sources to row sources or dimensions in the Integrated Operational Planning database.

statistical forecast Forecast data generated with ForecastPro and displayed in Oracle Integrated Operational Planning, Fusion Edition.

submit The action of saving/committing data to the data base.

XML See Extensible Markup Language.

Index

Symbols

[\\$add\(lhs,rhs\) function, 270](#)
[\\$begin function](#)
 (calendar) , [270](#)
 (calendar,begin-level,date,offset-level,offset), [270](#)
 (calendar,level), [271](#)
 (calendar,level,date), [271](#)
[\\$beginmember\(qname,offset\) function, 271](#)
[\\$coalesce\(arg1,...,argN\) function, 271](#)
[\\$date\(string,format\) function, 272](#)
[\\$datetime function](#)
 (calendar,level), [272](#)
 (calendar,level,date), [272](#)
[\\$lowercase\(field\) function, 274](#)
[\\$map\(field,test?a:b\) function, 274](#)
[\\$mondaymonth function, 275](#)
[\\$month function](#)
 \$month, [275](#)
 (calendar), [275](#)
 (calendar,date), [275](#)
[\\$mul\(lhs,rhs\) function, 275](#)
[\\$parent\(dimension,hierarchy,namespace,member\)](#)
 function, [276](#)
[\\$quarter function, 273](#)
 (calendar), [273](#)
 (calendar,date), [273](#)
 (date), [273](#)
[\\$quartername function](#)
 (calendar), [274](#)
 (calendar,date), [274](#)
[\\$rowsource\(rowsource,column,key1,expr1,](#)
 ...,keyN,exprN) function, [276](#)
[\\$rsubstr\(string,rbegin,rend\) function, 276](#)
[\\$seq function, 277](#)
[\\$seq\(name\) function, 277](#)
[\\$subfield\(field,delimiterChar,index\) function, 277](#)
[\\$substr\(s,begin,end\) function, 277](#)
[\\$today function, 278](#)

[\\$trim\(expr\) function, 278](#)
[\\$uppercase\(expr\) function, 278](#)
[\\$week function](#)
 (calendar), [278](#)
 (calendar,date), [279](#)
 (date), [279](#)
 week, [278](#)
[\\${name} function, 269](#)

A

[abs function, 189](#)
[add function, 189](#)
[administration workbench](#)
 about, [24](#)
 sections, [77](#)
[administrator, 22](#)
[all function, 220](#)
[allocation maps](#)
 creating or editing, [107](#)
 defining cube, [94](#)
[alter](#)
 alter model schema set current edit, [146](#)
 alter sandbox function, [157](#)
 alter system command function, [160](#)
[analysis](#)
 analysis history, viewing, [39](#)
 owner, [18](#)
 process, [15](#)
 roles
 administrator, [22](#)
 approver, [21](#)
 data provider, [21](#)
 participant, [20](#)
 types
 about, [111](#)
 adding key assumptions to, [114](#)
 adding key metrics to, [113](#)

- creating, [112](#)
- deleting, [115](#)
- viewing a scenario as, [70](#)
- viewing or editing, [115](#)
- viewing scenarios for, [30](#)
- workbench, description, [27](#)
- anchor cell in Excel, [56](#)
- and function, [189](#)
- Anteros
 - Ambassador Active-X Control, [286](#)
 - batch files, [164](#)
- application resources, managing, [136](#)
- approver, [21](#)
- assumptions, adding to analysis types, [114](#)
- attributeMap functions, [195](#)
- attributeReverseMap functions, [196](#)
- Avg function, [84](#)
- avg function
 - (double[] addends), [189](#)
 - (double[] addends, boolean includeEmptyCells), [189](#)

B

- basic functions, [188](#)
- block cache, [281](#)
- boolean literals, [172](#)

C

- calendarMap functions, [194](#)
- ceiling function, [189](#)
- changing models
 - after publishing, [109](#), [120](#)
- childMembers function, [203](#)
- children function
 - (hierarchy), [203](#)
 - (location hierarchy), [203](#)
 - (member, hierarchy), [204](#)
- clear schema shadow, [147](#)
- committing plan changes, [38](#)
- comparing scenarios, [35](#)
- connections
 - adding, [127](#)
 - deleting, [128](#)
- constraints
 - assigned, [112](#)
 - creating, [97](#)

- defining formulas for, [182](#)
- context menus, [58](#)
- copying scenarios, [35](#)
- cos functions, [199](#)
- Count function, [83](#)
- count function, [204](#)
- cousinLocations function
 - (past time members), [204](#)
 - (root member), [204](#)
- create sandbox, [157](#)
- creating scenarios, [31](#)
- cube function, [205](#)
- cube-to-cube maps, [99](#)
- cubes, creating, [93](#)
- current function, [220](#)
- currentTime function, [227](#)
- currentTimeMember function, [227](#)
- currentValue function, [205](#)
- custom
 - functions, [233](#)
 - measures sets, [65](#)

D

- daily scripts, [167](#)
- data collection
 - process, [17](#)
 - scenarios, [41](#)
- data providers
 - about, [41](#)
 - assigning to data collection scenarios, [42](#)
 - monitoring status, [43](#)
 - sending workbooks to, [42](#)
 - workflow, [21](#)
- data source, creating, [87](#)
- data source-to-row source maps, [102](#)
- datasouce-to-dimension maps, [105](#)
- dateMemberProperty function, [205](#)
- dateProperty function, [205](#)
- deleting scenarios, [35](#)
- descendants function, [205](#)
- dimension
 - function, [205](#)
 - hierarchy literals, [173](#)
 - literals, [173](#)
- dimensionMember function, [206](#)
- dimensions
 - collapsing, [60](#)

- creating, [92](#)
- dense, [282](#)
- expanding, [60](#)
- searching, [61](#)
- worksheet dimensions, [53](#)
- div function, [189](#)
- doubleProperty function, [206](#)
- doubleValue function, [190](#)

E

- e-mail reader, enabling and disabling, [161](#)
- e-mailing
 - an Excel shortcut, [40](#)
 - scenarios, [40](#)
- editing scenarios, [32](#)
- eq function, [190](#)
- equals function, [206](#)
- Essbase connections, [127](#)
- exactMap functions, [197](#)
- Excel
 - installing the Anteros Ambassador Active-X control for, [286](#)
 - opening a scenario in, [52](#)
 - troubleshooting, [287](#)
- exceptions
 - about, [46](#)
 - viewing, [46](#)
- exists function, [206](#)
- exp functions, [200](#)
- export commands, [152](#)
- exporting
 - and importing models
 - from the Model tab, [109](#)
 - from the Presentation tab, [120](#)
 - models
 - from the Model tab, [110](#)
 - from the Presentation tab, [120](#)
 - multidimensional data, [162](#)
- external functions, [232](#)

F

- factorial(int x) functions, [200](#)
- favorites, adding scenarios to, [40](#)
- filters
 - adding a custom search filter, [45](#)
 - filtering a list, [45](#)

- first function, [220](#)
- First_in_Period function, [84](#)
- firstChild function
 - (hierarchy), [207](#)
 - (hierarchy, location), [206](#)
 - (hierarchy, member), [207](#)
- firstDescendant function, [207](#)
- fixed exceptions, [46](#)
- floor function, [190](#)
- forecasts
 - managing overrides, [75](#)
 - managing rules, [74](#)
 - reviewing, [73](#)
- formulas
 - functions, [187](#)
 - handling null cells, [183](#)
 - keywords, [171](#)
 - literals, [172](#)
 - operators, [175](#)
 - variables, [174](#)
- formulas, defining for
 - constraints, [182](#)
 - measures, [178](#)
- functions, [187](#)
 - \$add(lhs,rhs), [270](#)
 - \$begin, [270](#)
 - (calendar), [270](#)
 - (calendar,begin-level,date,offset-level,offset), [270](#)
 - (calendar,level), [271](#)
 - (calendar,level,date), [271](#)
 - \$beginmember(qname,offset), [271](#)
 - \$coalesce(arg1,...,argN), [271](#)
 - \$date(string,format), [272](#)
 - \$datetime
 - (calendar,level), [272](#)
 - (calendar,level,date), [272](#)
 - \$lowercase(field), [274](#)
 - \$map(field,test?a:b), [274](#)
 - \$mondaymonth, [275](#)
 - \$month
 - \$month, [275](#)
 - (calendar), [275](#)
 - (calendar,date), [275](#)
 - \$mul(lhs,rhs), [275](#)
 - \$parent(dimension,hierarchy,namespace,member), [276](#)

- \$quarter
 - \$quarter, [273](#)
 - (calendar), [273](#)
 - (calendar,date), [273](#)
 - (date), [273](#)
- \$quartername
 - (calendar), [274](#)
 - (calendar,date), [274](#)
- \$rowsource(rowsource,column,key1,expr1,
...,keyN,exprN), [276](#)
- \$rsubstr(string,rbegin,rend), [276](#)
- \$seq
 - \$seq, [277](#)
 - \$seq(name), [277](#)
- \$subfield(field,delimiterChar,index), [277](#)
- \$substr(s,begin,end), [277](#)
- \$today, [278](#)
- \$trim(expr), [278](#)
- \$uppercase(expr), [278](#)
- \$week
 - \$week, [278](#)
 - (calendar), [278](#)
 - (calendar,date), [279](#)
 - (date), [279](#)
- \${name}, [269](#)
- abs, [189](#)
- add, [189](#)
- all, [220](#)
- and, [189](#)
- attributeMap, [195](#)
- attributeReverseMap, [196](#)
- Avg, [84](#)
- avg
 - (double[] addends), [189](#)
 - (double[] addends, boolean includeEmptyCells),
[189](#)
- basic, [188](#)
- calendarMap, [194](#)
- ceiling, [189](#)
- childMembers, [203](#)
- children
 - (hierarchy), [203](#)
 - (location hierarchy), [203](#)
 - (member, hierarchy), [204](#)
- cos, [199](#)
- Count, [83](#)
- count, [204](#)
- cousinLocations
 - (past time members), [204](#)
 - (root member), [204](#)
- cube, [205](#)
- current, [220](#)
- currentTime, [227](#)
- currentTimeMember, [227](#)
- currentValue, [205](#)
- custom, [233](#)
- dateMemberProperty, [205](#)
- dateProperty, [205](#)
- descendants, [205](#)
- dimension, [205](#)
- dimensionMember, [206](#)
- div, [189](#)
- doubleProperty, [206](#)
- doubleValue, [190](#)
- eq, [190](#)
- equals, [206](#)
- exactMap, [197](#)
- exists, [206](#)
- exp, [200](#)
- external, [232](#)
- factorial(int x), [200](#)
- first, [220](#)
- First_in_Period, [84](#)
- firstChild
 - (), [207](#)
 - (hierarchy, location), [206](#)
 - (hierarchy, member), [207](#)
- firstDescendant, [207](#)
- floor, [190](#)
- futureSiblingsCount, [228](#)
- gcd, [200](#)
- ge, [190](#)
- getDateForCurrentTimeMember, [228](#)
- getPositive, [190](#)
- gt, [190](#)
- hierarchy, [207](#)
- hierarchyLevel, [207](#)
- in, [207](#)
- indexOf, [208](#)
- inflatedValue, [200](#)
- isAncestor, [208](#)
- isBatchMode, [208](#)
- isBlank, [190](#)
- isCurrent

- (current location), [228](#)
- (member), [228](#)
- (supplied location), [228](#)
- isEndOfPeriod, [229](#)
- isFuture
 - (current location), [229](#)
 - (member), [229](#)
 - (supplied location), [229](#)
- isInRange, [208](#), [229](#)
- isInRangeFromCurrentTime, [229](#)
- isLeaf, [208](#)
- isLeftRightLeaf, [230](#)
- isLevel, [208](#)
- isLeveln, [209](#)
- isMember, [209](#)
- isNext
 - (), [230](#)
 - (member), [230](#)
 - (supplied location), [230](#)
- isNull, [191](#)
- isNullString, [191](#)
- isPast
 - (current location), [231](#)
 - (member), [231](#)
 - (supplied location), [231](#)
- isPrevious
 - (), [231](#)
 - (member), [231](#)
 - (supplied location), [232](#)
- isRoot
 - (member), [209](#)
 - (member, hierarchy), [209](#)
- isStartOfPeriod, [232](#)
- isZero, [191](#)
- lag, [221](#)
- last, [221](#)
- Last_in_Period, [84](#)
- lastChild
 - (hierarchy), [209](#)
 - (location, hierarchy), [210](#)
 - (member, hierarchy), [210](#)
- lastDescendant, [210](#)
- le, [191](#)
- lead, [221](#)
- level
 - (level), [222](#)
 - (level, one attribute), [221](#)
 - (level, two attributes), [221](#)
- leveln, [222](#)
- levels, [222](#)
- location, [210](#)
- log, [200](#)
- log10, [200](#)
- lookup
 - (one key column), [211](#)
 - (two key columns), [211](#)
 - (three key columns), [211](#)
 - (four key columns), [211](#)
- lookupDate
 - (one key column), [212](#)
 - (two key columns), [212](#)
 - (four key columns), [212](#)
 - (three key columns), [212](#)
- lookupString
 - (one key column), [213](#)
 - (two key columns), [213](#)
 - (four key columns), [213](#)
 - (three key columns), [213](#)
- lt, [191](#)
- mapping, [194](#)
- math, [199](#)
- Max, [83](#)
- max
 - (double a, double b), [191](#)
 - (double[] mbrs), [191](#)
- member, [202](#)
 - (dimension), [214](#)
 - (location, dimension), [214](#)
- memberName
 - (dimension), [214](#)
 - (location, dimension), [214](#)
 - (member), [214](#)
- memberProperty, [215](#)
- Min, [84](#)
- min
 - (a, b), [192](#)
 - (double[] mbrs), [192](#)
- mod, [200](#)
- moduloNext, [222](#)
- moduloPrevious, [222](#)
- mul, [192](#)
- ne, [192](#)
- next, [222](#)
- nextMember

- (level), 215
- (member, level), 215
- nonLeaves, 222
- not, 192
- nval, 192
- offset, 223
- offsetFromCurrent, 232
- oneToOneMap, 197
- or, 192
- overlapCalendarMap, 195
- overlapMembers, 232
- parent
 - (hierarchy), 215
 - (location, hierarchy), 215
 - (member, hierarchy), 216
- parentMembers, 216
- parents, 216
- pastCousinLocations
 - (ancestor member), 216
 - (root member), 216
- pastSiblingsCount, 232
- pct, 193
- pctof, 193
- power, 201
- previous, 223
- previousMember
 - (Level level), 217
 - (member, level), 217
- property, 217
- random, 193
- range
 - (from an anchor member), 218
 - (from, to), 223
 - (member from, member to), 218
 - (member from, member to, level level), 218
 - (zero, to), 223
- Reverse, 235
- round, 193
- RowSourceLookup, 236
- rowsourceLookup, 197
- rowsourceQuery, 198
- RSQLGenerateSet, 235
- shape, 220
- siblingCount
 - (heirarchy), 218
 - (location, hierarchy), 219
- siblingLocations

- (hierarchy), 219
- (location, hierarchy), 219
- (location, hierarchy, boolean), 219
- siblings
 - (hierarchy), 223
 - (location, hierarchy), 224
- sin, 201
- singleMember, 224
- splice, 199
- split, 199
- sqrt, 201
- streq, 193
- string, 193
- stringValue, 193
- sub, 193
- Sum, 84
- sum, 194
- SystemPeriod(level), 236
- tan, 201
- time, 227
- timeMemberMap
 - (dimension), 195
 - (dimension, member), 195
- timeRange, 232
- toBoolean, 194
- toDouble, 194
- futureSiblingsCount function, 228

G

- gcd functions, 200
- ge function, 190
- generate commands, 153
- getDateForCurrentTimeMember function, 228
- getPositive function, 190
- groups
 - adding users and groups, 137
 - managing users and groups, 137
 - searching for users and groups, 138
 - viewing users and groups, 138
- gt function, 190

H

- hierarachy level literals, 174
- hierarchies
 - dynamic, 92
 - row source based, 92

- static, [92](#)
- hierarchy function, [207](#)
- hierarchyLevel function, [207](#)
- home page, [25](#)

I

- impact of plan changes, [68](#)
- import commands, [147](#)
- importing models
 - from the Model tab, [111](#)
 - from the Presentation tab, [120](#)
- in function, [207](#)
- in-line reports, [67](#)
- indexOf function, [208](#)
- inflatedValue functions, [200](#)
- introduced exceptions, [46](#)
- invoke command, [159](#)
- isadmin command-line tool
 - command options, [139](#)
 - commands
 - for building hierarchies, [150](#)
 - for loading members into dimensions, [150](#)
 - common activities, [140](#)
 - common usage, [140](#)
 - custom java code commands, [162](#)
 - export commands, [152](#)
 - exporting multidimensional data, [162](#)
 - generate commands, [153](#)
 - import commands, [147](#)
 - JACL scripts, [158](#)
 - load commands, [154](#)
 - macro commands, [145](#)
 - process flow modeling, [164](#)
 - publish commands, [148](#)
 - sandbox calculation, submit, and reconcile
 - commands, [157](#)
 - schema commands, [146](#)
 - script templates, [158](#)
 - system commands, [160](#)
 - using, [143](#)
 - xml files and search path, [142](#)
 - xml schema files, [142](#)
- isAncestor function, [208](#)
- isBatchMode function, [208](#)
- isBlank function, [190](#)
- isCurrent function
 - (current location) , [228](#)

- (member), [228](#)
- (supplied location), [228](#)
- isEndOfPeriod function, [229](#)
- isFuture function
 - (current location), [229](#)
 - (member), [229](#)
 - (supplied location), [229](#)
- isInRange function, [208](#), [229](#)
- isInRangeFromCurrentTime function, [229](#)
- isLeaf function, [208](#)
- isLeftRightLeaf function, [230](#)
- isLevel function, [208](#)
- isLeveln function, [209](#)
- isMember function, [209](#)
- isNext function
 - (), [230](#)
 - (member), [230](#)
 - (supplied location), [230](#)
- isNull function, [191](#)
- isNullString function, [191](#)
- isPast function
 - (current location), [231](#)
 - (member), [231](#)
 - (supplied location), [231](#)
- isPrevious function
 - (), [231](#)
 - (member), [231](#)
 - (supplied location), [232](#)
- isRoot function
 - (member), [209](#)
 - (member, hierarchy), [209](#)
- isStartOfPeriod function, [232](#)
- isZero function, [191](#)

J

- JACL scripts, [165](#)
- java code commands, [162](#)
- java heap size, [282](#)
- jobs
 - adding, [129](#)
 - deleting, [131](#)
 - editing details, [130](#)
 - executing, [130](#)
 - managing the job queue, [128](#)
 - viewing, [129](#)

K

- key assumptions
 - adding to analysis types, [114](#)
 - in the Excel Show Impact window, [69](#)
- key metrics
 - adding measures to, [64](#)
 - adding to analysis types, [113](#)
 - assigning key metrics to users, [113](#)
 - in the Excel Show Impact window, [69](#)

L

- lag function, [221](#)
- last function, [221](#)
- Last_in_Period function, [84](#)
- lastChild function
 - (hierarchy), [209](#)
 - (location, hierarchy), [210](#)
 - (member, hierarchy), [210](#)
- lastDescendant function, [210](#)
- le function, [191](#)
- lead function, [221](#)
- level function
 - (level), [222](#)
 - (level, one attribute), [221](#)
 - (level, two attributes), [221](#)
- leveln function, [222](#)
- levels function, [222](#)
- literals
 - boolean, [172](#)
 - dimension, [173](#)
 - dimension hierarchy, [173](#)
 - hierarchy level, [174](#)
 - member, [173](#)
 - number, [172](#)
 - string, [173](#)
- load commands, [154](#), [156](#)
- load replace dimensions, [151](#)
- load xml specifications, [237](#)
- location function, [210](#)
- log function, [200](#)
- log10 function, [200](#)
- logging in, [22](#)
- lookup function
 - (four key columns), [211](#)
 - (one key column), [211](#)
 - (three key columns), [211](#)
 - (two key columns), [211](#)

- lookupDate function
 - (four key columns), [212](#)
 - (one key column), [212](#)
 - (three key columns), [212](#)
 - (two key columns), [212](#)
- lookupString function
 - (four key columns), [213](#)
 - (one key column), [213](#)
 - (three key columns), [213](#)
 - (two key columns), [213](#)
- lt function, [191](#)

M

- macro commands, [145](#)
- mail reader, enabling and disabling, [161](#)
- mapping and data flow, [85](#)
- mapping functions, [194](#)
- maps
 - allocation, [107](#)
 - creating, [99](#)
 - cube-to-cube, [99](#)
 - data source-to-dimension, [105](#)
 - data source-to-row source, [102](#)
 - row source-to-cube, [101](#)
- math functions, [199](#)
- Max function, [83](#)
- max function
 - (double a, double b), [191](#)
 - (double[] mbrs), [191](#)
- MDX extensions, [235](#)
- measures
 - adding to key metrics, [64](#)
 - changing values, [64](#)
 - creating, [94](#)
 - custom measure sets, [65](#)
 - editing, [96](#)
 - Excel formula to display value changes, [67](#)
 - expanding and collapsing, [61](#)
 - formula about, [83](#)
 - formulas for, [178](#)
 - level-dependent measure sets, [66](#)
 - recalculating values, [64](#)
 - searching, [61](#)
 - selecting for display, [63](#)
 - summarization, [83](#)
 - worksheet measures, [54](#)
- member function

- (dimension), [214](#)
- (location, dimension), [214](#)
- member functions, [202](#)
- member literals, [173](#)
- memberName function
 - (dimension), [214](#)
 - (location, dimension), [214](#)
 - (member), [214](#)
- memberProperty function, [215](#)
- merging scenarios, [34](#)
- migration scripts, [168](#)
- Min function, [84](#)
- min function
 - (a, b), [192](#)
 - (double[] mbrs), [192](#)
- mod functions, [200](#)
- Model tab,
 - understanding the user interface, [86](#)
- models
 - building, [87](#)
- moduloNext function, [222](#)
- moduloPrevious function, [222](#)
- mul function, [192](#)

N

- ne function, [192](#)
- next function, [222](#)
- nextMember function
 - (level), [215](#)
 - (member, level), [215](#)
- nonLeaves function, [222](#)
- not function, [192](#)
- null cells, [183](#)
- number literals, [172](#)
- nval function, [192](#)

O

- offset function, [223](#)
- offsetFromCurrent function, [232](#)
- onChange instructions, [176](#)
- oneToOneMap functions, [197](#)
- or function, [192](#)
- overlapCalendarMap functions, [195](#)
- overlapMembers function, [232](#)

P

- parent function
 - (hierarchy), [215](#)
 - (location, hierarchy function), [215](#)
 - (member, hierarchy), [216](#)
- parentMembers function, [216](#)
- parents function, [216](#)
- participant, [20](#)
- pastCousinLocations function
 - (ancestor member), [216](#)
 - (root member), [216](#)
- pastSiblingsCount function, [232](#)
- pct function, [193](#)
- pctof function, [193](#)
- Perl module scripts, [167](#)
- planning workbench
 - about, [23](#)
 - pages, [25](#)
- planning workbooks
 - understanding, [53](#)
- worksheet
 - cells, [55](#)
 - dimensions, [53](#)
 - measures, [54](#)
- power functions, [201](#)
- previous function, [223](#)
- previousMember function
 - (Level level), [217](#)
 - (member, level), [217](#)
- printing scenarios, [39](#)
- process flow modeling, [164](#)
- profile, updating, [28](#)
- property function, [217](#)
- publish commands, [148](#)
- publishing
 - changing models after publishing, [109](#), [120](#)
 - objects in the Model tab, [108](#)
 - objects in the Presentation tab, [119](#)

Q

- queries
 - about, [116](#)
 - creating, [116](#)
 - deleting, [118](#)
 - previewing, [117](#)

R

random function, [193](#)
 range function
 (from an anchor member), [218](#)
 (from, to), [223](#)
 (member from, member to), [218](#)
 (member from, member to, level level), [218](#)
 (zero, to), [223](#)
 reconcile commands, [157](#)
 report templates
 adding, [118](#)
 understanding, [118](#)
 viewing in Excel, [119](#)
 reports
 in-line reports, [67](#)
 viewing, [50](#)
 reset worksheet view, [56](#)
 resources, managing, [136](#)
 Reverse function, [235](#)
 round function, [193](#)
 row source
 based hierarchies, [92](#)
 creating, [89](#)
 row source-to-cube maps, [101](#)
 RowSourceLookup function, [236](#)
 rowsourceLookup function, [197](#)
 rowsourceQuery function, [198](#)
 RSQLGenerateSet function, [235](#)

S

sandbox calculation commands, [157](#)
 scenarios
 accessing from the Home page, [29](#)
 adding to favorites, [40](#)
 approving, [38](#)
 committing plan changes, [38](#)
 comparing, [35](#)
 completing, [38](#)
 copying, [35](#)
 creating, [31](#)
 data collection, [41](#)
 deleting, [35](#)
 e-mailing, [40](#)
 e-mailing an Excel shortcut, [40](#)
 editing, [32](#)
 merging change entries, [34](#)
 opening in Excel, [52](#)

 printing, [39](#)
 refreshing data, [71](#)
 reviewing changes, [68](#)
 submitting for approval, [36](#)
 undoing changes, [33](#)
 viewing, [29](#)
 as another analysis type, [70](#)
 for specific analysis types, [30](#)
 in the Analysis Workbench, [30](#)
 schema commands, [146](#)
 script files
 adding, [131](#)
 deleting, [132](#)
 viewing, [132](#)
 script templates, [158](#)
 adding, [133](#)
 deleting, [134](#)
 editing, [134](#)
 managing, [133](#)
 scripts
 daily, [167](#)
 in install-root/bin, [164](#)
 JACL, [165](#)
 managing, [71](#)
 migration, [168](#)
 Perl module, [167](#)
 weekly, [168](#)
 security filters
 adding, [135](#)
 deleting, [136](#)
 editing, [135](#)
 set schema shadow, [146](#)
 shape functions, [220](#)
 siblingCount function
 (heirarchy), [218](#)
 (location, hierarchy), [219](#)
 siblingLocations function
 (hierarchy), [219](#)
 (location, hierarchy), [219](#)
 (location, hierarchy, boolean), [219](#)
 siblings function
 (hierarchy), [223](#)
 (location, hierarchy), [224](#)
 sin functions, [201](#)
 singleMember function, [224](#)
 snapshot command, [162](#)
 solvers, [162](#)

- splice functions, [199](#)
- split functions, [199](#)
- sqrt functions, [201](#)
- stage command, [155](#)
- stage replace, [151](#)
- statistical forecasts
 - managing override, [75](#)
 - managing rules, [74](#)
 - reviewing, [73](#)
- streq function, [193](#)
- string function, [193](#)
- string literals, [173](#)
- stringValue function, [193](#)
- sub function, [193](#)
- submit commands, [157](#)
- submit sandbox, [158](#)
- Sum function, [84](#)
- sum function, [194](#)
- system commands, [160](#)
- system resources, managing, [136](#)
- SystemPeriod(level) function, [236](#)

T

- tan functions, [201](#)
- tasks
 - creating, [48](#)
 - deleting, [50](#)
 - editing, [49](#)
 - viewing, [47](#)
- templates, script, [71](#), [133](#)
- time
 - functions, [227](#)
 - varying, [90](#)
- timeMemberMap functions
 - (dimension), [195](#)
 - (dimension, member), [195](#)
- timeRange function, [232](#)
- toBoolean function, [194](#)
- toDouble function, [194](#)

U

- undoing scenario changes, [33](#)
- users
 - adding users and groups, [137](#)
 - managing users and groups, [137](#)
 - searching for users and groups, [138](#)

- updating profile, [28](#)
- viewing users and groups, [138](#)
- users and groups
 - managing, [137](#)

V

- viewing
 - scenarios, [30](#)

W

- weekly scripts, [168](#)
- workbooks
 - about, [121](#)
 - creating, [121](#)
 - deleting, [122](#)
 - sending to data providers, [42](#)
- worksheets
 - cells, [55](#)
 - changing layout, [56](#)
 - context menus, [58](#)
 - creating, [123](#)
 - deleting, [126](#)
 - dimensions, [53](#)
 - editing, [126](#)
 - hiding, [125](#)
 - measures, [54](#)
 - toolbar and menu options, [56](#)
 - understanding, [122](#)
 - viewing details, [125](#)

X

- XML
 - files
 - cube definitions, [86](#)
 - data source definitions, [85](#)
 - dimension definitions, [86](#)
 - row source definitions, [86](#)
 - schema, [142](#)
 - search path, [142](#)
 - stagemap, [85](#)
 - load XML specification, [237](#)
 - planning worksheet template files, [123](#)

